Treball de Fi de Grau

**ENGINYERIA EN TECNOLOGIES INDUSTRIALS**

# Control d'un manipulador redundant basat en un model dinàmic obtingut per identificació

**MEMÒRIA**


**Autor:** Antoni Planells Valencia
**Directora:** Carme Torras Genís


**Convocatòria:** Juliol 2014




Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona


*Agraïments*

*Aquest Treball de Fi de Grau ha estat una experiència molt enriquidora i gratificant que no hauria estat possible sense l'ajut, ànim i recolzament de molta gent. Per això voldria fer arribar el meu agraïment a tots els que l'han fet possible, i en particular*

*a la meva directora la Dra Carme Torras per haver dipositat en mi la seva confiança, per haver-me acollit dins del seu grup donant-me l'oportunitat de gaudir fent una mica de recerca i per estar accessible sempre que ho he necessitat,*

*al Dr Carlos Ocampo i al doctorant Adrià Colomé pel seu constant i incondicional ajut. Els seus consells i ensenyances de segur que em seran molt de profit en el meu desenvolupament com a enginyer,*

*a tot l'equip de l'IRI pel seu tracte i per facilitar-me el treball oferint-me el seu suport quan m'ha fet falta,*

*als meus pares, a la meva germana i a la meva família pel seu recolzament diari, per la seva paciència i pels seus consells,*

*als meus amics pels seus ànims i la seva companyia.*

# Resum

Aquest projecte, que s'ha dut a terme a l'Institut de Robòtica i Informàtica Industrial, s'ha desenvolupat en l'àmbit de l'automàtica i el control. En ell, s'ha treballat amb el manipulador redundant WAM, un braç robòtic de 7 graus de llibertat, que presenta com a característica la seva lleugeresa pel fet de tenir els seus motors a la base i aplicar els moviments mitjançant cables interns. Aquesta característica fa que aquest robot pugui ser utilitzat en entorns en els que interactui amb l'ésser humà.

En concret, s'han aplicat estratègies de control i d'aprenentage basades en el model dinàmic del robot que han permès controlar-lo de manera precisa i robusta. L'abast del projecte ha anat, doncs, des de l'estudi inicial del problema fins al control del robot i la validació dels controladors dissenyats.

S'han executat múltiples experiments per obtenir un conjunt de dades significatiu i representatiu de la dinàmica que experimenta el robot en el seu moviment. Aquestes dades s'han tractat amb el software de càlcul MATLAB mitjançant tècniques d'identificació paramètriques per obtenir models de les dinàmiques de cadascuna de les articulacions del robot, que s'han validat posteriorment amb els indicadors KPIs i el percentatge d'aproximació del model respecte de la realitat. Després de l'etapa de modelització s'ha aplicat el control predictiu basat en models com a estratègia a utilitzar per aconseguir un conjunt de controladors per al seguiment precís de les trajectòries executades pel robot. En aquesta etapa s'han fet servir programes d'optimització del MATLAB per minimitzar les funcions objectiu definides per al nostre problema, respectant les restriccions físiques i els rangs d'operació del robot. La validació dels controladors dissenyats s'ha fet comparant les trajectòries originals amb les executades gràcies al control predictiu.

Finalment s'ha fet un estudi dels costos i de l'impacte mediambiental del projecte, juntament amb algunes reflexions sobre el que podria ser una futura línia d'estudi com a continuació d'aquest projecte.

# Índex

# 1 Definició de variables i paràmetres

| *Variable/paràmetre* | *Definició* | *Unitats* |
|---|---|---|
| $\theta_{real}(t)$ | Vector posició angular real | rad |
| $\dot{\theta}_{real}(t)$ | Vector velocitat angular real | rad/s |
| $\ddot{\theta}_{real}(t)$ | Vector acceleració angular real | rad/s$^2$ |
| $\theta_{des}(t)$ | Vector posició angular desitjat | rad |
| $\dot{\theta}_{des}(t)$ | Vector velocitat angular desitjat | rad/s |
| $\ddot{\theta}_{des}(t)$ | Vector acceleració angular desitjat | rad/s$^2$ |
| $\tau$ | Vector parell aplicat a les articulacions | N·m |
| $e_{pos}(t)$ | Error de posició en una trajectòria al llarg del temps | rad |
| $M(\theta(t))$ | Tensor d'inèrcia dinàmic | kg·m$^2$ |
| $C(\theta(t), \dot{\theta}(t))$ | Forces centrípetes i de Coriolis durant el moviment | N·m |
| $F_{fric}(t)$ | Parell aplicat a les articulacions degut a la fricció | N·m |
| $F_{grav(t)}$ | Parell aplicat a les articulacions degut a la gravetat | N·m |
| $T$ | Període de mostratge | s |
| $u(t)$ | *Inputs* dels models de velocitat | N · m |
| $y(t)$ | *Outputs* dels models de velocitat | rad/s |
| $T_p$ | Període d'oscil·lació dels parells d'excitació del robot | s |
| $\omega$ | Pulsació de mostratge del robot | rad/s |
| $\omega_p$ | Pulsació d'oscil·lació dels parells d'excitació del robot | rad/s |
| $N$ | Nombre de mostres per període d'oscil·lació en una trajectòria | |
| MSE | Mitjana dels errors de posició al quadrat | rad$^2$ |
| MAE | Mitjana del mòdul de l'error de posició | rad |

# 2 Prefaci

## 2.1 Origen del projecte

L'estiu passat vaig tenir el privilegi de ser seleccionat pel Consejo Superior de Investigaciones Científicas (CSIC) com a becari d'un projecte d'iniciació a la investigació per estudiants universitaris (JAE INTRO 2013) a l'Institut de Robòtica i Informàtica Industrial de Catalunya (IRI) sota la direcció de la Dra. Carme Torras Genís. D'aquesta manera vaig entrar a formar part del grup de *Percepció i Manipulació* com a *undergraduate student*. Citant textualment la web del grup, "la investigació del grup de *Percepció i Manipulació* es centra en la millora de la percepció, l'aprenentatge, i les capacitats de planificació dels robots per assolir un major grau d'autonomia i facilitat d'ús en tasques de manipulació quotidianes. Alguns dels temes abordats són la interpretació geomètrica de la informació perceptual, la construcció de models d'objectes 3D, selecció de l'acció i la planificació, aprenentatge per reforç, i l'ensenyament per demostració". Per aquests estudis, en el Laboratori de Percepció i Manipulació es treballa amb una sèrie de braços robot (dos robots Staübli i dos robots WAM), amb diversos robots humanoides educacionals així com amb càmeres de color i profunditat i sensors de força. L'IRI és un centre d'investigació capdavanter en Robòtica i treballar i aprendre amb ells ha estat molt important per la meva formació com a futur Enginyer Industrial.

Durant aquesta beca vaig treballar en el projecte [6] titulat *Identificació i test del model dinàmic d'un braç robot* i que va ser presentat com a treball dirigit en el meu currículum del Grau en Enginyeria en Tecnologies Industrials. El projecte consistia en treballar amb el robot WAM (*Whole Arm Manipulator*), un braç robòtic de 7 graus de llibertat, que es caracteritza pel fet de ser lleuger al tenir els motors a la base i aplicar moviments mitjançant cables interns. Es tractava d'obtenir una bona aproximació de les friccions que actuen sobre el robot mitjançant experiments i de deduir els factors que hi influeixen. Després s'havia d'utilizar aquesta aproximació en un model dinàmic complet del robot, implementar-lo en el robot i validar-lo amb una altra sèrie d'experiments. Pel seu desenvolupament, en tot moment vaig estar assessorat i guiat, a més a més de per la Dra. Carme Torras, pel Dr. Carlos Ocampo i el doctorant Adrià Colomé.

Més concretament, el projecte va consistir en modelitzar els parells de fricció que apareixien a les articulacions del braç robòtic durant el seu moviment per tot l'espai. L'expressió matemàtica d'aquest model obtingut juntament amb algunes imatges il·lustratives es veuran a l'apartat 4.4 d'aquesta memòria.

El parell que s'aplica a les articulacions del robot per donar-li moviment és el resultat de sumar tres termes que corresponen a un controlador PID, a un model de les forces de Coriolis i de les forces d'inèrcia i a un model de la fricció, [1]. El fet de tenir un terme que correspon al model de la fricció treu pes al terme que correspon al controlador, i això es tradueix en fer més petites les constants del controlador cosa que fa que les trajectòries tinguin un comportament menys rígid.

Abans de la realització d'aquest primer projecte, el robot WAM tenia un sistema d'aprenentatge LWPR (*Locally Weighted Projection Regression*)[8] ,que era específic per a cada trajectòria; és a dir, que primer s'executava una trajectòria amb un controlador PID ja implementat i després es calculava el model de la fricció per a aquella trajectòria en concret.

Aquest mètode oferia un bon nivell d'aproximació per a aquesta trajectòria però el caràcter era molt local i per tant només s'aconseguia el comportament amigable per ella. Per a qualsevol altra trajectòria el comportament continuava essent rígid i poc interactuable amb els humans.

Amb el projecte en el qual vaig treballar es va aconseguir un model de fricció general per a cada una de les articulacions del braç robot (7 en total) que no depenia de les trajectòries executades.

L'obtenció d'aquest model va resultar ser de gran utilitat ja que a partir d'aquell moment es va poder implementar un nou controlador PID amb les constants molt més petites que feia que el braç robòtic pogués interactuar amb els humans fàcilment, sense suposar un perill.

En efecte, quan s'aturava una trajectòria del robot utilitzant el controlador rígid calia fer molta força i per això, un cop deixàvem el braç robòtic lliure, aquest recuperava la seva posició en la trajectòria movent-se a gran velocitat i amb tanta força que el feia perillós. Amb el controlador tou en canvi, aturar el robot requereix molt menys esforç i el robot és capaç de tornar a la trajectòria desitjada de forma suau.

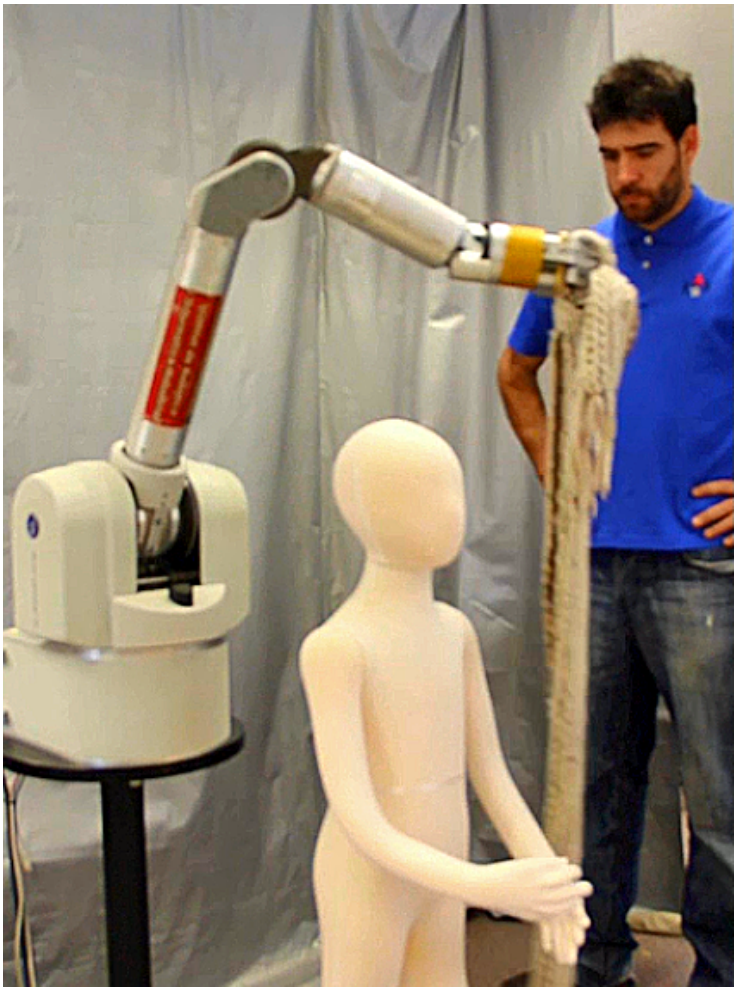

Figura 1: Robot WAM del Laboratori de Percepció i Manipulavió de l'IRI.

Quan vàrem acabar el projecte ens vàrem preguntar per altres tècniques de control avançat aplicables a aquest problema, la seva implementació en el braç robòtic i la compleció del model introduint nous termes corresponents a la seva dinàmica.

Aquest és l'origen del nou projecte que vull presentar com a Treball Final de Grau i que s'ha realitzat al mateix laboratori de Percepció i Manipulació de l'IRI sota la direcció de la doctora Carme Torras i el suport constant del doctor Carlos Ocampo i del doctorant Adrià Colomé.

Es pot considerar aquest nou projecte com la continuació natural del primer, però, per la seva major complexitat, també molt més ambiciós.

## 2.2 Motivació

La motivació per treballar en aquest projecte ha estat doble:

Des d'un punt de vista personal, com ja he dit anteriorment, treballar a l'IRI és un privilegi. En el projecte anterior l'experiència va ser molt productiva i enriquidora. Vaig treballar i aprendre molt en temes que no es tracten a la carrera i vaig aprofundir en d'altres de coneguts. També vaig aprendre molt en formes de treball, vaig interactuar amb investigadors del centre i em vaig enfrontar a continus reptes i entrebancs molt engrescadors. Per tant, continuar l'experiència em motivava molt.

Des d'un punt de vista científic també penso que el projecte proposat és molt interessant. La robòtica és una ciència en constant evolució i actualització que cada cop té més pes en la vida quotidiana dels humans, per la qual cosa el fet d'explorar i estudiar noves tècniques de control aplicables a robots per a fer-los més amigables resulta de gran interès. En aquest sentit es presenta aquest projecte com un petit gra de sorra que vol contribuir en l'evolució d'aquesta ciència.

## 2.3 Requeriments previs

La robòtica combina moltes disciplines, com per exemple: l'àlgebra, el càlcul, la física, la mecànica, l'electrònica, la informàtica, la intel·ligència artificial, la enginyeria de control, etc. Encara que la recerca en robòtica es subdivideix en moltes sublínies diferents, es tracta d'una ciència que requereix el domini de coneixements de diferents disciplines, i fa que l'investigador hagi de ser força interdisciplinar. La necessitat d'aquest maduresa segurament fa que la robòtica en els estudis d'enginyeria sigui una branca més pròpia de màsters que no de graus.

Com ja he dit, les disciplines que combina la robòtica són moltes i encara que per al projecte que he treballat han estat de gran utilitat els coneixements obtinguts en assignatures com ara Dinàmica de Sistemes, Control Automàtic, Mecànica, Àlgebra lineal o Geometria, entre d'altres, les meves mancances també eren importants.

Una primera feina va ser doncs obtenir una base teòrica suficient per poder entendre de manera correcta el funcionament cinemàtic i dinàmic dels robots amb els que anava a treballar. Per això, vaig haver d'estudiar diversos capítols del llibre [7]. Amb ell,

vaig començar per entendre la robòtica en termes generals, les seves aplicacions i a aprendre a diferenciar els diversos tipus de robots en funció de la seva estructura i forma (forma de braç, vehicle, etc.) que condicionen directament la seva utilitat i funcionalitat. Després vaig estudiar la cinemàtica d'aquests robots amb un tractament analític complex fent servir canvis de base i de referències a l'espai, translacions i rotacions a l'espai, translacions homogènies i angles d'Euler, entre d'altres tècniques d'anàlisi cinemàtica. Més endavant vaig aprendre els conceptes de cinemàtica diferencial i estàtica fent referència bàsicament al procediment d'obtenció de la cinemàtica inversa. La cinemàtica inversa és la funció que determina la posició angular de les articulacions necessària per situar l'efector final del robot en una certa posició/orientació de l'espai cartesià, i que pot tenir vàries solucions (habitualment infinites) per a robots redundants com el WAM. Per últim, vaig estudiar l'anàlisi dinàmica i energètica dels robots amb diversos graus de llibertat amb un tractament matemàtic i matricial força complex.

Per als diversos càlculs, simulacions i creacions de programes i *scripts*, he profunditzat en la utilització del software de càlcul MATLAB que ja havia fet servir durant la carrera. D'altra banda, per dur a terme l'execució de certes trajectòries i la validació dels controladors he hagut d'adquirir nocions bàsiques en C++, [10] i [18], llenguatge per a mi desconegut fins al desenvolupament d'aquest projecte. Voldria també comentar que he hagut d'aprende a escriure aquesta memòria amb el processador de text LaTeX[2], ja que està especialment orientat a la creació de llibres, documents científics i tècnics que continguin fòrmules matemàtiques per als quals els processadors de text estàndards són molt ineficients.

El gruix d'aquests aprenentatges el vaig fer durant el projecte previ a aquest Treball de Fi de Grau. Això i l'experiència que ja havia adquirit a l'IRI m'ha estat de gran ajut i ha agilitzat moltes de les tasques.

# 3 Introducció

## 3.1 Objectius i abast del projecte

L'objectiu principal d'aquest projecte és l'aplicació d'estratègies de control i d'aprenentatge basades en el model dinàmic del robot que ens permetin controlar-lo de manera més precisa i robusta que fins ara.

Quan es diu que es vol controlar de forma més precisa i robusta el robot es vol dir que l'objectiu és aconseguir que el robot sigui capaç de seguir qualsevol trajectòria que se li ordeni amb les mínimes desviacions i que aquestes tinguin la menor magnitud possible.

El robot experimenta alguns fenòmens, com la fricció dels cables durant el moviment, que introdueixen incertesa, per la qual cosa no és possible executar una trajectòria exactament igual que la desitjada. Per tant, com més bons siguin els models i els controladors millor serà el seguiment de les trajectòries. El càlcul de certs KPI (*Key Perfomance Index*) i l'estudi de l'anomenat percentatge d'aproximació (%fit) de les trajectòries seran un bons indicadors a l'hora de fer les validacions corresponents.

Per això s'obtindrà primer un model dinàmic que aproximi bé la realitat dinàmica del robot i després es dissenyaran els controladors que es creguin adients per aconseguir el control del robot.

Per aconseguir l'objectiu desitjat es seguiran els passos que tot seguit s'especifiquen:

- Definició correcta del problema i estudi del sistema a tractar.
- Obtenció del model dinàmic del robot.
- Validació del model.
- Disseny dels controladors apropiats.
- Validació dels controladors implementats.

Es farà l'estudi de la dinàmica del robot seguint els passos anteriors per a cadescuna de les articulacions, 7 en total, que s'anomenaran objectius parcials. Això es farà per tal de reduir la dimensió dels programes amb què treballarem, però sobretot perquè hi ha certes

parelles d'articulacions que presenten acoblament entre els seus motors, raó per la qual el seu comportament requereix una atenció especial.

En una primera part, l'obtenció dels models i el disseny dels controladors es faran utilitzant el software de càlcul Matlab que és el que es fa servir per defecte a l'IRI. Però després, per executar les trajectòries i validar els controladors es treballarà amb programes escrits en el llenguatge de programació C++, que és el codi en el que estan els executables i compiladors que permeten realitzar operacions amb el robot.

L'abast del nostre projecte va, doncs, des de l'estudi inicial del problema fins al control del robot i una posterior validació dels controladors dissenyats.

## 3.2 Metodologia de treball

La metodologia de treball utilitzada en aquest projecte ha anat variant en funció de l'etapa del projecte en la que s'estava treballant i dels objectius parcials que s'estaven intentant assolir. En aquest apartat es farà un resum de la metodologia aplicada en aquestes etapes, encara que una descripció més exhaustiva es trobarà en els capítols de la memòria dedicats a cada una d'elles.

La primera etapa va consistir en l'estudi del sistema i el plantejament del problema.

Es va fer un procés de recerca d'informació i es van llegir alguns llibres i articles per poder entendre bé el funcionament del robot WAM [7], saber quines accions se li podien encomanar i com se'l podia controlar. En aquesta primera etapa també es van definir els objectius de control que es volien assolir en el nostre sistema amb el qual es treballarà. Per això, es van entendre els fenònems més rellevants que afectaven a la cinemàtica i a la dinàmica del robot. En aquest estudi, es van entendre correctament les variables amb les que s'ha treballat i els rangs de valors en els que es podrien moure. Aquesta informació serà clau per no sobredimensionar el problema que es vol resoldre. Un cop acabada aquesta etapa es va tenir una idea clara de la dimensió del nostre problema i de com caldria abordar-lo per mirar de resoldre'l.

Un cop plantejat el problema i estudiat el sistema en detall, es van dur a terme les etapes corresponents a l'obtenció dels models dinàmics i les seves posteriors validacions, una per cada articulació.

La metodologia de treball aplicada en aquestes etapes va ser molt similar i, per tant, s'explicaran conjuntament. El que s'ha fet ha estat l'obtenció de dos paquets de dades, un per obtenir els models i l'altre per validar-los, i s'han tractat adientment amb el programa Matlab. Les dades s'han pres executant un seguit de trajectòries mitjançant excitacions que nosaltres introduíem al sistema i que hem pogut controlar en tot moment. En cap moment s'ha fet ús d'un controlador per calcular les dades de parell a introduir al robot, ja que el que es volia era fer un experiment en llaç obert sense realimentar el sistema per així observar com aquest responia davant d'excitacions diverses. Les dades que s'han pres corresponen a les posicions, a les velocitats i als parells aplicats a les diverses articulacions en cada instant per a cada trajectòria.

Un cop preses les dades, s'ha passat a treballar amb la llibreria d'identificació paramètrica de models del Matlab anomenada *System Identification Toolbox* [12], que és una de les més utilitzades per a realitzar tasques de modelització.

Mitjançant aquesta llibreria s'han anat obtenint diferents models, des de models lineals simples fins a models no lineals complexos, amb l'objectiu de buscar un nivell d'aproximació el més gran possible. Per triar el model més precís a l'hora d'aproximar les dades, s'ha fet ús d'indicadors KPIs com ara la mitjana dels errors al quadrat (MSE) o la mitjana dels errors en valor absolut (MAE) i el percentatge d'aproximació (% de fit) que calcula el *System Identification Toolbox*.

Depenent de l'articulació que es volia modelitzar, el model ha treballat amb diferents *inputs* però amb els mateixos *outputs*. En concret, les articulacions que no presenten acoblament amb cap altra (la 1, la 4 i la 7) han presentat models on els *inputs* eren els parells aplicats a les articulacions i els *outputs* la velocitat a la que es movia l'articulació. En canvi les dues parelles d'articulacions amb acoblament mutu entre els seus motors (les articulacions 2-3 i les articulacions 5-6) han presentat models on els *inputs* eren els parells aplicats a les dues articulacions de la parella i on els *outputs* eren la velocitat a la que es movia l'articulació corresponent. El nivell d'aproximació obtingut ha estat similar per a cada articulació. La validació d'aquests models ha consistit en introduir les dades de parell, utilitzades per a executar certes trajectòries, en el model i calcular quines velocitats de sortida donava. Després, s'han comparat aquestes velocitats amb les velocitats reals a les que s'ha mogut el robot. També s'han comparat les posicions per on s'ha mogut el

robot amb les posicions integrades a partir de les velocitats que calculava el model.

Un cop obtinguts els models s'ha passat a treballar en temes de control i en concret en el control predictiu basat en models, [4].

En aquesta etapa s'ha continuat treballant amb el Matlab però s'ha passat a utilitzar una llibreria d'optimització anomenada *Optimization Toolbox* [13], que ens ha permès seguir les trajectòries. Amb aquesta llibreria i més en concret amb la funció `fmincon`, [14], s'ha pogut definir la funció objectiu que es volia aconseguir per garantir que el seguiment de les trajectòries es dugués a terme mitjançant uns parells aplicats a les articulacions que no presentessin variacions altes i així evitar fer malbé el robot. Aquest programa d'optimització utilitzava els models obtinguts per calcular les velocitats a les que s'hauria de moure el robot en cada instant per seguir la trajectòria desitjada, sempre complint les restriccions físiques i els rangs operacionals que s'han especificat en la definició del problema. Els models utilitzats per mirar de determinar quin parell aplicar en cada instant s'han anomenat *models de predicció* mentres que els models que s'han utilitzat per simular el comportament del robot (planta del nostre sistema) s'han anomenat *models de simulació* (que poden ser els mateixos). D'aquesta manera s'ha aconseguit tancar el sistema en llaç tancat aconseguint controladors predictius que utilitzaven els models que s'han obtingut mitjançant la identificació paramètrica que garanteixen que el comportament de la planta del nostre sistema sigui l'esperat. S'ha fet una validació d'aquests controladors, introduint els valors de parell calculats a partir del programa d'optimització del Matlab directament al robot, mirant la trajectòria que el robot executava i comparant-la amb la trajectòria que realment havia executat el robot amb els parells reals. La validació implementant en codi C++ tots els programes d'optimització i els models directament a l'ordinador del WAM, s'ha deixat per un futur donat que la seva complexitat requeria molt més temps del que disposàvem.

La resta d'etapes del projecte han estat dedicades a la planificació temporal del projecte, al càlcul dels costos associats, a l'estudi de l'impacte ambiental que s'ha pogut produir, i a pensar en futures línies de treball que aquest projecte suggereix.

## 3.3 Alguns entrebancs en el projecte

Com en tot projecte ambientat en l'àmbit de la recerca, a vegades alguns resultats no surten com un s'espera i s'ha de mirar de refer els camins que inicialment s'estaven seguint per intentar complir amb les especificacions que s'hagin concretat. El nostre projecte de recerca ambientat en la robòtica no ha estat una excepció i ens hem anat trobant amb alguns entrebancs que s'han anat superant. Per exemple, el procediment realitzat en les etapes de modelització o de disseny de l'estructura dels controladors no ha estat fàcil i, de fet, es van provar inicialment altres alternatives a les que consten en aquesta memòria que van donar pitjors resultats, per la qual cosa van ser rebutjades.

Però també s'han donat problemes tècnics al llarg del projecte. Es va començar treballant amb un dels robot WAM del laboratori amb el qual es van fer les diferents modelitzacions de cadascuna de les articulacions. Però quan s'estava acabant l'etapa de modelització i validació va haver-hi problemes tècnics, aliens al nostre projecte, que van deixar incapacitat el robot per molt de temps. Això va suposar un dur entrebanc per a nosaltres ja que es van haver de realitzar de nou les tasques de modelització amb el segon robot WAM que hi havia al laboratori. Es podria pensar que tractant-se del mateix tipus de robot WAM no caldria recalcular tots els models, però no va ser així ja que les dinàmiques dels dos robots diferien una mica ja sigui, per exemple, per la tensió dels seus cables o pel funcionament dels seus motors interns.

Un cop obtinguts els nous models amb aquest segon robot, es van dissenyar els controladors predictius que es desitjava implementar. Aleshores, durant l'etapa de validació un dels cables interns del robot es va trencar i va quedar inutilitzat el segon robot durant uns dies.

De tots aquests entrebancs tècnics se'n pot extreure una lectura positiva i és que s'ha vist que els dos robots es poden modelar amb models de la mateixa estructura, encara que amb paràmetres propis diferents, i es poden controlar amb controladors de funcionaments equivalents. El projecte realitzat és aplicable doncs als dos robots WAM del laboratori.

## 3.4 Estructura de la memòria

A continuació es farà una breu descripció de les seccions en què, a partir d'aquest moment, està dividida la memòria.

A la Secció 4 d'aquesta memòria es donarà la definició del problema juntament amb una descripció detallada del sistema amb el qual s'ha treballat. També es definiran els objectius de control que es volien assolir en aquest projecte i es descriuran els elements del nostre sistema de control. Per altra banda, es comentaran les restriccions físiques i els rangs operacionals que prenen les variables del nostre sistema. Per últim, es parlarà de les incerteses i de les pertorbacions del sistema, com ara la fricció dinàmica que pateix el robot en el seu moviment.

A la Secció 5 s'explicarà la modelització dinàmica del robot. Primer, es presentarà la metodologia de treball que s'ha seguit en aquesta etapa del projecte. Després, es comentaran i es justificaran els diferents experiments que s'han anat realitzant. Tot seguit, s'especificarà quins indicadors s'han utilitzat per a avaluar la qualitat dels models que s'anaven obtenint. Es descriuran, a continuació, els models obtinguts tant analíticament, explicitant les seves equacions, com gràficament, mitjançant figures obtingudes amb el MATLAB. Finalment, es mostraran els resultats obtinguts en la validació dels models.

A la Secció 6 es parlarà de l'optimització i el control que s'ha dut a terme en el projecte. Primer, s'explicaran les tècniques de control que s'han aplicat per tal de complir amb els objectius especificats. A continuació, es comentarà el procés d'optimització que s'ha dut a terme mitjançant la llibreria d'optimització *Optimization Toolbox* del Matlab. Tot seguit, es detallaran els resultats obtinguts amb aquesta optimització. Per últim, s'explicarà el procés de validació que s'ha aplicat per validar els controladors que s'han dissenyat.

A la Secció 7 es farà una quantificació dels costos que ha suposat el desenvolupament d'aquest projecte, dividits en diverses partides com els derivats dels recursos humans, de l'amortització de l'equip, els costos generals i els beneficis industrials. Finalment es sumaran tots els costos parcials per determinar el cost total del projecte.

A la Secció 8 es farà una avaluació de l'impacte mediambiental que ha suposat el nostre projecte.

A la Secció 9 es plantejarà una futura línia d'estudi que es podria desenvolupar a partir de la feina feta en aquest projecte.

A la Secció 10 es presentaran les conclusions del projecte tenint en compte els objectius plantejats i els resultats que s'han obtingut, a més a més de fer alguna consideració personal.

A la Secció *Bibliografia* es relacionarà la bibliografia consultada al llarg del projecte i que s'ha referenciat en aquesta memòria.

A la Secció d'*Annexos* es mostraran, en primer lloc, alguns resultats teòrics rellevants fets servir en l'elaboració d'aquest projecte. A continuació es mostraran els codis tant dels programes en C++ com els de MATLAB que s'han utilitzat en algun moment del projecte. Per últim, es mostraran les taules dels indicadors obtinguts en la modelització de les diverses articulacions del robot.

# 4 Definició del problema

## 4.1 Descripció del sistema

El sistema amb el qual s'ha treballat és el robot WAM, un braç robòtic de 7 graus de llibertat, que es caracteritza per ser lleuger pel fet de tenir els seus motors a la base i aplicar els moviments mitjançant cables interns. Aquest robot implementa canvis en la concepció tradicional de manipuladors, incorporant variacions en els sistemes mecànics i en els dispositius de control i realimentació de les articulacions. Tot això comporta avantatges com la disminució de la fricció i del consum d'energia, així com una millora de la *back-drivability* (resposta a forces externes) que fa que el robot sigui un dispositiu segur per a la interacció amb els humans.


Figura 2: Braç robot WAM. Imatge extreta de www.barrett.com

El comportament d'aquest robot es regeix segons una equació dinàmica que descriu el seu moviment i que presenta la forma següent:

$$M(\theta(t)) \cdot \ddot{\theta}(t) + C(\theta(t), \dot{\theta}(t)) + F_{fric}(t) + F_{grav}(t) = \tau_c(t) - \tau_e(t).$$

Els termes $\theta(t)$, $\dot{\theta}(t)$ i $\ddot{\theta}(t)$ corresponen a les posicions, velocitats i acceleracions de les diferents articulacions del braç. El terme $M(\theta(t))$ correspon al tensor d'inèrcia que

depèn de la posició del robot en tot moment. El terme $C(\theta(t), \dot{\theta}(t))$ correspon als parells centrípets i de Coriolis que apareixen durant el moviment del robot i que depenen de la posició i de la velocitat angular del robot en tot moment. Els termes $F_{fric}(t)$ i $F_{grav}(t)$ corresponen als parells de fricció i de gravetat presents també durant el moviment del robot i que són els termes que tenen més rellevància dintre de l'equació. El parell de fricció apareix degut al frec dels cables interns del robot i és difícil de modelitzar amb equacions o funcions matemàtiques. Respecte al parell de gravetat, el robot presenta una opció anomenada *compensació de gravetat* que permet situar el robot en qualsevol posició de l'espai quiet a l'espera d'executar alguna trajectòria. Per això, a vegades, s'omet aquest terme en l'equació del moviment. Per últim, els termes $\tau_c(t)$ i $\tau_e(t)$ corresponen als parells aplicats a les articulacions mitjançant un controlador o per una acció exterior. Normalment no s'aplica cap tipus d'acció exterior sobre el robot durant les seves trajectòries a no ser que es tingui un control molt robust sobre aquest que li permeti recalcular les trajectòries un cop el fem desviar mitjançant accions externes. De moment, doncs, el terme $\tau_e(t)$ es considerarà negligible i, per tant, es reescriurà el terme $\tau_c(t) - \tau_e(t)$ com a $\tau(t)$. L'equació dinàmica del moviment del robot s'escriurà doncs:

$$M(\theta(t)) \cdot \ddot{\theta}(t) + C(\theta(t), \dot{\theta}(t)) + F_{fric}(t) + F_{grav}(t) = \tau(t). \tag{1}$$

Observem que com que aquesta equació inclou totes les articulacions, els termes $\theta(t)$, $\dot{\theta}(t)$, $\ddot{\theta}(t)$ i $\tau(t)$ són vectors de $\mathbb{R}^7$ on cada component correspon a una articulació en concret.

De vegades, segons el que es vulgui estudiar, convé més treballar amb models obtinguts de l'experimentació que permetin determinar la velocitat i/o la posició a partir dels parells aplicats que treballar amb l'equació (1). Aquesta modelització pot ser lineal o no lineal, però ha de garantir un model que aproximi de manera precisa la realitat dinàmica del robot.

Abans de parlar de models convé explicar una mica més com funciona el robot internament i quins *inputs* necessita per seguir una trajectòria. Per tal que el robot segueixi correctament una determinada trajectòria es necessita que en cada instant se

li introdueixin les dades corresponents als parells a aplicar a les diferents articulacions. Aquests són els unics *inputs* del robot. Els parells provenen d'unes tensions aplicades als motors de les articulacions i que guarden una relació proporcional amb ells amb unes constants pròpies que té el robot. Aquestes constants així com altres detalls estan reflectits en el manual del WAM [11].

Aquests parells aplicats provenen d'un controlador que pot utilitzar altres dades per calcular-los en tot instant que poden correspondre a les posicions o velocitats que es desitja que el robot tingui. La seva manipulació o comparació amb altres dades com les posicions o velocitats reals del robot permeten la implementació de controladors PIDs que compensin els errors de posició o velocitat durant les trajectòries. D'aquesta manera es pot parlar que els parells són el conjunt de variables que permeten excitar el nostre sistema ja que són els encarregats de fer moure el robot.

Per últim, cal dir que aquest robot treballa amb un període de mostratge de 0.02s i per tant cada 0.02s se li poden introduir noves ordres per garantir un correcte seguiment de les trajectòries.

Un cop conegut l'únic *input* del nostre robot, podem començar a pensar en quin model dinàmic obtenir i amb quines variables d'entrada treballarà i quines variables de sortida o *outputs* tindrà.

En un principi es va pensar en un model que ens permetés obtenir les velocitats de les articulacions a partir dels parells aplicats. D'aquesta manera, si obteníem una bona aproximació de la velocitat, afegint un element integrador en el nostre sistema, juntament amb el coneixement de la posició inicial del robot seríem capaços de reproduir trajectòries de manera fidel i precisa. El model, per tant, haurà de ser una equació de la següent forma:

$$\dot{\theta}(t) = f(\tau(t)) \tag{2}$$

Aleshores, mitjançant un integrador i el coneixement de la posició inicial del robot es podrà obtenir el valor de posició per a qualsevol instant de temps $t_0$:

$$\theta(t_0) = \theta_{inicial} + \int_0^{t_0} \dot{\theta}(t)\, dt = \theta_{inicial} + \int_0^{t_0} f(\tau(t))\, dt, \tag{3}$$

amb $\theta_{inicial} = \theta(t = 0)$.

Com que els controladors treballen en temps discret cal reescriure l'equació (2) i (3) de la següent manera:

$$\dot{\theta}(KT + T) = f(\tau(KT)) \tag{4}$$

i

$$\theta(KT) = \theta_{inicial} + \sum_{k=0}^{K} \dot{\theta}(kT)T = \theta_{inicial} + \sum_{k=0}^{K} f(\tau(kT))T,$$

amb $t_0 = KT$ i on $T = 0.02$s és el període de mostratge del robot.

Pot sobtar que el parell aplicat a l'instant $KT$ permeti calcular la velocitat a l'instant $KT + T$, però això es deu a la causalitat del sistema i al fet que si no fos així el sistema seria estàtic i no dinàmic. Si un sistema no és dinàmic això vol dir que el sistema no pot variar d'estat i evolucionar, cosa que difereix totalment de la realitat del robot.

En un principi es difícil intuir la complexitat del model dinàmic que aproxima millor la realitat, raó per la qual en aquest projecte s'anirà provant diversos models des de lineals senzills fins no lineals més complexos. Això es veurà amb detall en la determinació dels models utilitzant tècniques d'identificació paramètriques [3].

## 4.2 Objectiu i sistema de control

L'objectiu principal d'aquest projecte és aconseguir un control robust i precís de les trajectòries executades pel robot que a més li facin tenir un comportament tou i amigable davant d'accions externes.

El robot de per si ja té implementats una sèrie de controladors que poden dur a terme les tasques de control de les trajectòries amb un comportament més o menys rígid. En aquest projecte, però, es presenta una estratègia diferent de control anomenat control predictiu que consisteix en una família d'estratègies de control que fan ús dels models per conèixer l'evolució de les seves dinàmiques i complir els objectius de control proposats mitjançant la minimització (per optimització) d'una funció de cost. El control predictiu

és la millor opció per controlar un procés amb múltiples entrades i sortides tot satisfent les restriccions d'operació del propi sistema. Aquesta tècnica és una tècnica moderna de control que ha guanyat popularitat en els últims anys i que es detallarà a l'apartat 6.1.

Per tal de garantir un seguiment precís de les trajectòries que es vulguin executar amb el robot, el nostre objectiu de control es podria descriure de la següent manera:

$$\lim_{t\to+\infty} e_{pos}(t) := \lim_{t\to+\infty} (\theta_{des}(t) - \theta(t)) = 0$$

on $e_{pos}(t)$ indica l'error de posició en l'espai d'articulacions(error de posició angular de cada articulació), $\theta_{des}(t)$ la posició desitjada i $\theta(t)$ la posició real.

Volem doncs obtenir un control de la posició que minimitzi l'error de posició definit com la diferència entre les posicions angulars de cada articulació corresponents a les trajectòries desitjades i les corresponents a les trajectòries reals en els diferents instants de mostratge.

Recordem que el model que es vol aconseguir aproximarà la velocitat del robot i que la posició s'aconseguirà mitjançant un element integrador col·locat en sèrie amb el model. Caldrà doncs posar èmfasi en que el model aproximi de manera realista la velocitat de les articulacions; és a dir, fent que les difèrencies entre les velocitats reals i les aproximades pel model siguin mínimes en els instants de mostratge. Per controlar la posició mitjançant un control de la velocitat també caldrà disposar de informació sobre la posició inicial del sistema.

El sistema de control amb el qual es treballarà comptarà amb els elements bàsics d'un sistema de control. El robot WAM serà la nostra planta i s'intentarà modelitzar el seu comportament. Aquesta planta tindrà per entrades $u(t)$ els parells a aplicar a les articulacions i per sortides $y(t)$ les seves velocitats. A partir d'ella i de les entrades que se li apliquin es podran executar els experiments en llaç obert que es desitgin.

Com que el nostre objectiu és controlar en llaç tancat l'error de posició, caldrà afegir elements al sistema de control. En primer lloc, s'afegirà un element integrador després de la planta per poder integrar la velocitat i així obtenir la posició en qualsevol instant a partir del coneixement de l'estat inicial del sistema (posició inicial). Després s'afegirà el controlador abans de planta. El seu funcionament s'explicarà als apartats 6.2 i 6.3. Per

últim, el sistema haurà de presentar una realimentació de velocitat i no de posició ja que el model treballa amb velocitats.

A continuació es presenta una figura que mostra l'esquema de blocs d'un sistema de control amb la mateixa estructura que el nostre:

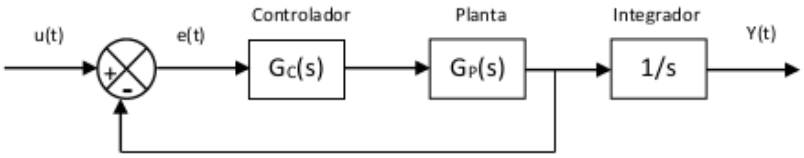


Figura 3: Esquema d'un sistema de control clàssic.

## 4.3 Rangs operacionals i restriccions físiques

El robot WAM té definit un cert espai de treball que compleix un conjunt de restriccions físiques i rangs operacionals. Per exemple, el rang de posicions que pot prendre qualsevol de les articulacions és un interval acotat fora del qual mai prendran valors les variables. Aquestes restriccions són degudes a la pròpia forma del robot i a la manera amb què s'ha construït la seva estructura articulada. El robot presenta també uns intervals de velocitats i acceleracions pels quals està dissenyat per treballar de forma correcta. Cal dir que els intervals en els que es poden moure les nostres variables no tenen perquè ser iguals en totes les articulacions del braç i, per tant, a l'hora de definir-los caldrà especificar en tot moment a quina articulació ens estem referint. També pot donar-se el cas que el rang d'algunes variables sigui igual per tota articulació.

Seguidament es detalla amb precisió els rangs de valors que poden prendre les variables $\theta(t)$, $\dot{\theta}(t)$, $\ddot{\theta}(t)$ i $\tau(t)$ ja que aquesta informació serà de molta utilitat quan es vulgui optimitzar i controlar el sistema mitjançant funcions d'optimització que es faran servir en els apartats 6.3, 6.4 i 6.5.

Recordant que $\theta(t) = (\theta_i(t))_{i=1:7}$ és un vector de $\mathbb{R}^7$ on cada component correspon a una articulació en concret, es té que els intervals on poden prendre valor les diverses posicions de les articulacions són els següents:

| | $\theta_1(t)$ | $\theta_2(t)$ | $\theta_3(t)$ | $\theta_4(t)$ | $\theta_5(t)$ | $\theta_6(t)$ | $\theta_7(t)$ |
|---|---|---|---|---|---|---|---|
| en rad | $(-2.6, +2.6)$ | $(-2.0, +2.0)$ | $(-2.8, +2.8)$ | $(-0.9, +3.1)$ | $(-4.8, +1.3)$ | $(-1.6, +1.6)$ | $(-2.2, +2.2)$ |
| en graus | (-150,+150) | (-113, +113) | (-157, +157) | (-50,+180) | (-275,+75) | (-90,+90) | (-128,128) |

El software amb el qual està implementat el WAM té un sistema de seguretat mitjançant el qual es poden interrompre trajectòries en qualsevol moment si es creu convenient. Les trajectòries del WAM es controlen mitjançant un comandament que té una sèrie de botons lluminosos que presenten els colors verd, groc i vermell que avisen si el sistema funciona correctament (verd), si el sistema es mou una mica ràpid i cal anar amb precaució (groc) o quan el braç adquireix velocitats massa grans i perilloses (vermell). Els valors pels quals s'encenen els botons es poden modificar tot canviant el codi del programa amb el què s'ha implementat el WAM, però com que són valors base determinats pels fabricants ja són una bona referència.

El botó groc s'il·lumina quan la velocitat d'alguna articulació supera els 0.5rad/s i s'activa el botó vermell quan aquesta velocitat arriba als 2rad/s, encara que per a major seguretat seria millor que no superés el 1rad/s. Per tant, anàlogament que per a les posicions, definirem els intervals de velocitat per a totes les articulacions com:

$$-2.0\text{rad/s} \leq \dot{\theta}_i(t) \leq +2.0\text{rad/s}, \qquad i = 1:7$$

Aquestes velocitats angulars a les articulacions es tradueixen en velocitats lineals en els eixos cartesians X, Y i Z a l'extrem final del braç de manera que si aquest adquireix certs nivells de velocitat els botons també s'encenen. El botó groc s'encén quan l'extrem del braç es mou a 0.5m/s i el botó vermell s'encen quan l'extrem es mou a 2m/s.

Pel que fa a les acceleracions en principi no hi ha cap valor límit imposat, però de totes maneres i per no superar les velocitats límits definides anteriorment seria convenient no superar mai els $2\text{rad/s}^2$. Per tant, anàlogament que per les posicions i les velocitats, definirem els intervals d'acceleració per a totes les articulacions com:

$$-2.0\text{rad/s}^2 \leq \ddot{\theta}_i(t) \leq +2.0\text{rad/s}^2, \qquad i = 1:7$$

Per últim, seria molt convenient acotar els límits inferiors i superiors dels valors que poden prendre els parells aplicats a cadascuna de les articulacions, ja que si s'introdueixen parells molt alts el robot es pot moure de manera brusca i vibrar de manera indeguda, cosa que pot danyar les articulacions i fer malbé el robot mateix. Els intervals pels parells aplicables a cadascuna de les articulacions depenen de cada articulació i són els següents:

| | $\tau_1(t)$ | $\tau_2(t)$ | $\tau_3(t)$ | $\tau_4(t)$ | $\tau_5(t)$ | $\tau_6(t)$ | $\tau_7(t)$ |
|---|---|---|---|---|---|---|---|
| en $N \cdot m$ | $(-8.0, +8.0)$ | $(-5.0, +5.0)$ | $(-3.5, +3.5)$ | $(-2.5, +2.5)$ | $(-0.6, +0.6)$ | $(-0.55, +0.55)$ | $(-0.2, +0.2)$ |

## 4.4 Incerteses i pertorbacions del sistema

Una incertesa en l'equació del moviment del WAM és el valor del parell de fricció que actua sobre el robot essent un fenomen molt significatiu que té un pes molt important com a parell aplicat a les articulacions. La seva modelització va ser l'objectiu de l'anterior projecte [6] realitzat també a l'IRI que tot seguit es resumirà.

En un principi es creia que les friccions eren només de tipus viscós però després d'alguns experiments es va veure que la fricció també era funció de la posició de les articulacions i que involucrava funcions no trivials:

$$F_{fric}(t) = f(\theta(t), \dot{\theta}(t))$$

A més de la dificultat que va suposar modelitzar la fricció, hi havia una font molt important de variabilitat en les dades recollides pel controlador PID que hi havia implementat. En efecte, el component derivatiu d'aquest controlador era tan elevat que feia que les dades presentessin un soroll afegit que dificultava el seu posterior tractament o anàlisi. Es va haver de realitzar, aleshores, un procés de filtratge de les dades per reduir tant com es pogués aquest soroll derivatiu, encara que mai no és possible obtenir una funció fonamental amb variabilitat nul·la.

A continuació s'exposarà, breument, com es va dur a terme el procés de modelització d'aquests parells de fricció juntament amb la seva posterior validació. També es comentarà la comparació que es va realitzar entre els models que es tenia antigament, que seguien el

procediment LWPR, i els nous models obtinguts. Per últim es comentaran les repercusions que va tenir l'obtenció d'aquest model nou de fricció.

En el projecte [6] es va obtenir un model general de la fricció per a cada una de les articulacións. Aquests parells eren particularment difícils d'aproximar ja que presentaven un cert comportament de cicle d'histèresi si es mirava el gràfic de $F_{fric}(t)$ en funció de $\dot{\theta}(t)$ i un comportament oscil·lant si es mirava el gràfic $F_{fric}(t)$ en funció $\theta(t)$. Per a determinar els diferents models es va dur a terme una metodologia de treball similar a la que s'utilitzarà en aquest projecte. Es van prendre dos paquets de dades de forma que un es va utilitzar per obtenir els models i l'altre per validar-los. Inicialment es partir d'una equació proposada a [5]. A l'hora de determinar la forma del model es va anar seleccionant funcions que es creien convenients per aproximar les dades i mitjançant un algoritme de mínims quadrats es calculaven els diferents pesos d'aquestes funcions. L'anàlisi de la mitjana dels errors al quadrat que es cometia en l'aproximació servia per decidir si una funció ajudava o no a millorar el model prèviament obtingut. D'aquesta manera es va arribar a determinar un conjunt de funcions matemàtiques que garantien una molt bona aproximació.

Els models de la fricció de cadascuna de les articulacions van presentar la següent forma:

$$\begin{aligned} F_{fric_i} \quad = \quad & b_1 \cdot \theta_i + b_2 \cdot \dot{\theta}_i + b_3 \cdot atan(s \cdot \dot{\theta}_i + z \cdot sign(\ddot{\theta}_i)) + b_4 \\ & +0.5 \cdot (1 + sign(\dot{\theta}_i)) \cdot (b_5 \cdot f_1 + b_6 \cdot f_2 + b_7 \cdot f_3 + b_8 \cdot f_4 + b_9 \cdot f_5) \\ & +0.5 \cdot (1 - sign(\dot{\theta}_i)) \cdot (b_{10} \cdot f_1 + b_{11} \cdot f_2 + b_{12} \cdot f_3 + b_{13} \cdot f_4 + b_{14} \cdot f_5), \quad i = 1:7. \end{aligned}$$

$$\text{on } f_j = \frac{4}{\pi} \cdot \frac{sin((2j-1) \cdot h \cdot \theta)}{2j-1}, \quad j = 1:5.$$

Per exemple, per l'articulació 1, el parell de fricció obtingut a partir del model ve representada gràficament de la següent manera, on Ffiltrada (color blau) correspon al parell de fricció real filtrat i Fricgeneral (color verd) correspon al parell de fricció calculat amb el model:

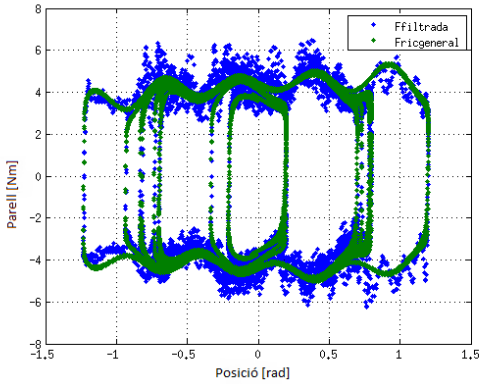


Figura 4: Parell de fricció en funció de la posició.

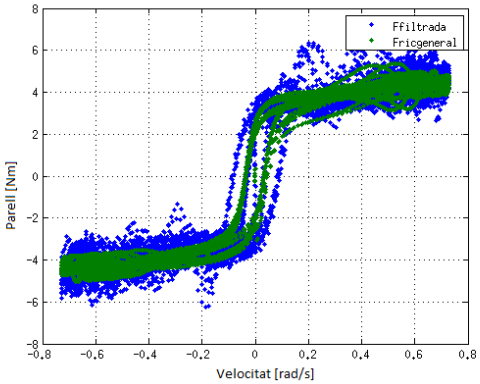


Figura 5: Parell de fricció en funció de la velocitat.

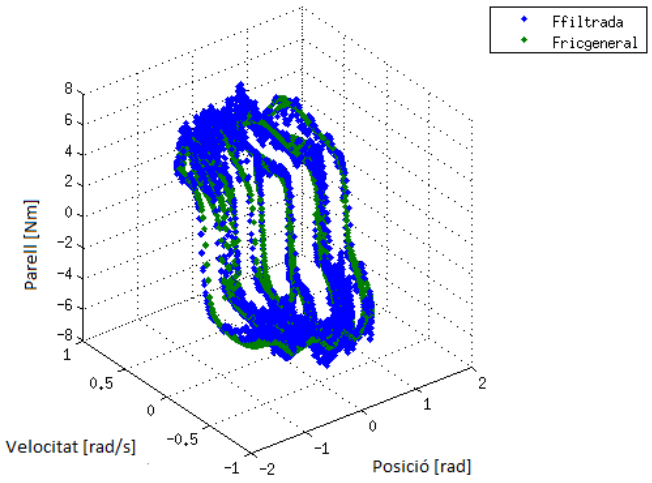


Figura 6: Parell de fricció en funció de la posició i de la velocitat.

Com es pot veure a les Figures 4, 5 i 6, és impossible aproximar les dades al cent per cent degut a la dispersió introduïda pel controlador PID, però també es pot observar que el nivell d'aproximació que aconsegueix el model és molt notable.

Un cop obtinguts els models amb el primer paquet de dades experimentals, es va procedir a la seva validació mitjançant el segon paquet de dades. Per això, es van introduir els *inputs* $\theta_i(t)$, $\dot{\theta}_i(t)$ i $\ddot{\theta}_i(t)$, $i = 1:7$, a cadascun dels models corresponents a les friccions de cada articulació, es va observar els valors dels parells que el model obtenia i se'ls va comparar amb els corresponents a la realitat.

A continuació es presenten, com a exemple, els resultats d'aquestes validacions corresponents al *joint* 1 prèviament modelitzat:

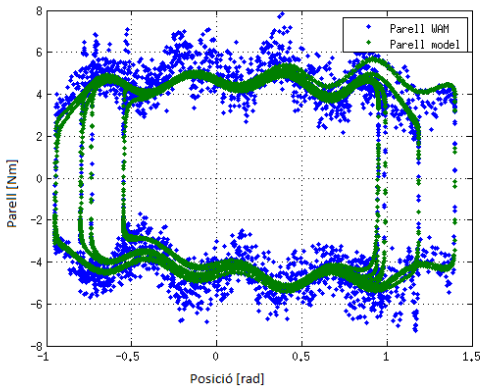


Figura 7: Parell de fricció en funció de la posició.

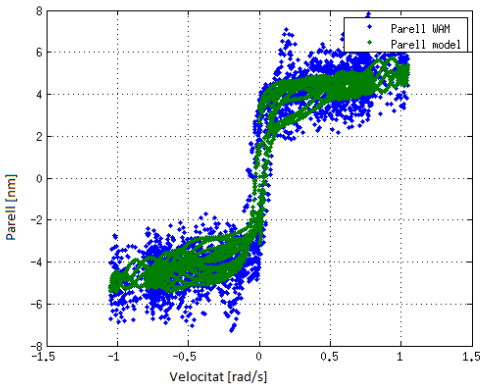


Figura 8: Parell de fricció en funció de la velocitat.

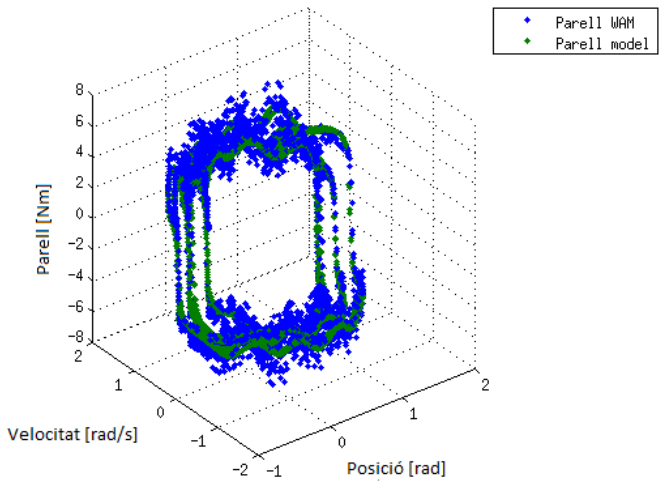


Figura 9: Parell de fricció en funció de la posició i de la velocitat.

Com es pot observar a les Figures 7, 8 i 9, tot i que les trajectòries del segon paquet de dades són diferents a les utilitzades per obtenir el model, l'aproximació de la fricció és molt bona malgrat la dispersió que presenten les dades.

Aquest procediment d'obtenció de models i la seva posterior validació es va dur a terme per a cadascuna de les articulacions del braç robot i es van aconseguir nivells similars d'aproximació global. D'aquesta manera es va obtenir un model global de la fricció a les articulacions al robot molt fidel a la realitat dinàmica d'aquest.

Un cop obtingut el model i validat es va dur a terme una comparació amb els models antics que es feien servir fins llavors i que eren obtinguts a partir de la tècnica LWPR. Aquests models aprenien el model de la fricció per a una trajectòria concreta i , per tant, tenien un caràcter molt local.

El procediment de comparació de models que es va dur a terme va ser el següent. Primer de tot es van definir un conjunt de trajectòries que el robot hauria de fer. A partir d'aquestes trajectòries es van definir altres trajectòries similars introduint petites variacions en els punts de recorregut. Aquests petits canvis haurien de accentuar el caràcter local dels models antics enfront del caràcter global del nou model que no hauria de variar la qualitat d'aproximació.

Un cop definides les diferentes trajectòries amb les seves diferents variants es va procedir a executar-les i després, les dades de posicions, de velocitats i d'acceleracions obtingudes es van utilitzar per determinar el valor de les friccions a través dels dos models. Com ja era d'esperar, els models antics perdien molta capacitat d'aproximació en introduir petites alteracions en la trajectòria inicialment definida. A continuació es presenten dues taules on es recullen les mitjanes dels errors al quadrat comesos pel model nou i pel model antic (LWPR), respectivament, corresponents a les friccions en les diferents articulacions per una trajectòria i les seves variants:

Articulacions en moviment: 1, 3, 4, 6, 7

Taula d'errors del nostre model:

| $e^2$ | Artic. 1 | Artic. 3 | Artic. 4 | Artic. 6 | Artic. 7 |
|---|---|---|---|---|---|
| Traj.1 | 0.7365 | 0.1847 | 0.1238 | 0.0098 | 0.0011 |
| Traj.1A | 0.6800 | 0.2126 | 0.1202 | 0.0121 | 0.0010 |
| Traj.1B | 0.7580 | 0.2213 | 0.1117 | 0.0160 | 0.0009 |
| Traj.1C | 0.7653 | 0.2110 | 0.1221 | 0.0128 | 0.0008 |
| Traj.1D | 0.7890 | 0.2288 | 0.1288 | 0.0189 | 0.0009 |
| Traj.1E | 0.8096 | 0.2190 | 0.1317 | 0.0175 | 0.0010 |

Taula d'errors del model LWPR:

| $e^2$ | Artic. 1 | Artic. 3 | Artic. 4 | Artic. 6 | Artic. 7 |
|---|---|---|---|---|---|
| Traj.1 | 0.9658 | 0.2569 | 0.1592 | 0.0188 | 0.0013 |
| Traj.1A | 1.5642 | 0.3258 | 0.2468 | 0.0213 | 0.0021 |
| Traj.1B | 2.6493 | 0.3329 | 0.1865 | 0.0259 | 0.0013 |
| Traj.1C | 2.2705 | 0.3292 | 0.2507 | 0.0210 | 0.0015 |
| Traj.1D | 1.7807 | 0.2732 | 0.2832 | 0.0203 | 0.0014 |
| Traj.1E | 2.9910 | 0.4294 | 0.2322 | 0.0353 | 0.0017 |

A continuació es presenta la comparació gràfica entre els dos models per l'articulació 3 corresponent a la trajectòria i les variants representades en les taules anteriors:

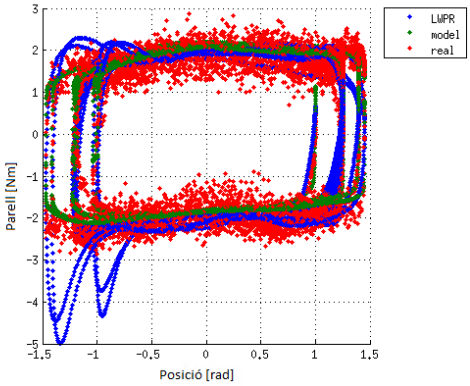


Figura 10: Comparació triple del parell de fricció.

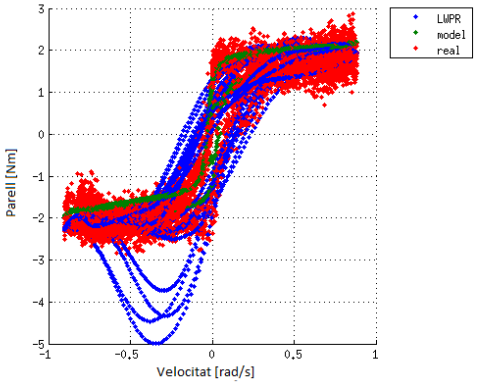


Figura 11: Comparació triple del parell de fricció.

Com es pot observar a les taules anteriors i les Figures 10 i 11, el guany d'aproximació aportat pels nous models és molt significatiu en comparació amb els models antics.

Un cop feta la comparació es va procedir a la implementació dels nous models en C++ per a poder-los fer servir en qualsevol moment que es requerís. Aquesta implementació va permetre després una posterior implementació d'un nou controlador PID amb constants molt més petites i que per tant permetia executar trajectòries amb el robot oferint un caràcter i resistència menys rígid de cara a accions exteriors i amb una capacitat de recuperar trajectòries un cop nosaltres el paràvem fent poca força. Aquest fet va ajudar molt en diversos temes com els relacionats amb tècniques d'aprenentatge de trajectòries del robot.

Com s'observa, aquest model de fricció és un model no lineal i això ens va fer sospitar que pel que fa a la modelització de la velocitat de les articulacions a partir dels parells, que és un dels objectius d'aquest projecte, aquest difícilment tindrà un caràcter purament lineal. Això es veurà de manera exhaustiva en els apartats 5.4 i 5.5 d'aquest projecte.

# 5 Modelització dinàmica del robot

## 5.1 Tècniques de modelització i metodologia emprada en el modelat i posterior validació

Per començar a parlar de la modelització dinàmica cal primer de tot definir com es faran els experiments. A continuació explicitarem les consideracions a tenir en compte:

- L'objectiu serà excitar el sistema mitjançant diferents parells d'entrada que variïn al llarg del temps i mirar com el sistema respon a aquestes entrades. Aquestes respostes del sistema són les que s'intentarà modelitzar. En el nostre cas, el que s'estudiarà és la velocitat a la que es va movent el robot. No s'ha estudiat com a resposta a modelitzar les posicions a les que es movia el robot, ja que no es van obtenir bons resultats d'identificació paramètrica dels models.

- Aquesta excitació del sistema es vol que sigui en llaç obert, és a dir sense realimentació del sistema i per tant els parells que s'apliquin a les articulacions no podran provenir de cap controlador que les hagi obtingut mitjançant cap tipus de càlcul. Aquest fet permetrà reduir també el soroll que podrien presentar les dades de parell si s'executessin mitjançant un controlador PID.

- Es definirà una expressió analítica de les entrades per a poder-les introduir-les al robot i que aquest es mogui.

- Variables com les posicions, les velocitats o les acceleracions desitjades no tindran cap incidència en el model ni en les trajectòries executades ja que aquestes respondran només a les excitacions que s'introdueixin al robot. El que sí s'haurà de tenir en compte és l'estat (posició) inicial del sistema.

Aquestes consideracions difereixen bastant del procediment que es va seguir en el projecte anterior per la modelització de la fricció. En aquell projecte s'executaven les trajectòries que seguien unes posicions, velocitats i acceleracions instantànies desitjades, que eren calculades internament amb uns *splines* que té el WAM implementats, i es calculaven els parells que calia aplicar en tot moment per seguir-les. Tot això es feia amb controladors PIDs que introduïen soroll en les dades corresponents als parells.

En el projecte actual el que es pretén és excitar el sistema amb dades de parells no contaminades per un controlador.

Per aconseguir-ho caldrà, d'una banda, substituir el controlador per un bloc constant o en el seu defecte treure'l directament del diagrama de blocs. Substituint o treient el controlador s'aconseguirà eliminar el soroll de les dades d'entrada corresponent als parells i a més a més es disposarà del control absolut a l'hora d'introduir-les. Un aspecte important és decidir de quina manera introduir les dades corresponents als parells i de quina manera s'organitzaran els experiments tant per a la modelització com per a la validació posterior. Per altra banda, no es disposarà de dades referents a trajectòries desitjades amb les seves posicions, velocitats i acceleracions ja que el que es vol és aplicar entrades al sistema i veure com aquest respon. La sortida del nostre model seguirà sent la velocitat a la que es mouen les articulacions i mitjançant procediments d'integració posteriors es podrà saber la trajectòria el robot.

Un cop introduïts aquests canvis passarem a realitzar experiments amb el nostre sistema (el robot WAM) en llaç obert de manera que tindrem constància, en tot instant, de les entrades amb les que s'excitarà el sistema i de les seves respostes.

En el que segueix s'explicarà com s'han organitzat els experiments i quines articulacions s'han anat movent. Primer, es modelarà una articulació lliure, com la 1 per exemple, i després una de les parelles d'articulacions que presenta acoblament mutu, com ara les articulacions 2-3. La modelització es durà a terme per separat, dividint els experiments en dos conjunts on en un conjunt es mou l'articulació 1 i en l'altre es mouen les articulacions 2 i 3. La modelització de l'articulació 1 es podrà estendre de forma similar per les articulacions 4 i 7 que també són lliures i, de la mateixa manera, si la modelització de la parella 2-3 a la de la parella 5-6. L'organització dels experiments de cara a la validació dels models serà la mateixa. Un tema igualment important és definir la manera d'introduir les dades de parells al robot de forma que amb múltiples experiments s'obtingui un conjunt significatiu i representatiu de dades de la realitat dinàmica que experimenta el robot en el seu moviment. S'ha pensat en aplicar senyals d'entrada pels parells de les articulacions en forma sinusoidal i anar variant l'amplitud i la freqüència d'aquests. Els parells d'entrada a qualsevol de les articulacions haurien de presentar una forma similar a la següent:

$$\tau_i(t) = \tau_{max_i} sin(\omega_p t), \quad t \geq 0, \quad i = 1:7.$$

L'equació reescrita en temps discret seria:

$$\tau_i(KT) = \tau_{max_i} sin(\omega_p KT), \quad K \geq 0, \quad i = 1:7\ , \tag{5}$$

on $\tau_{max_i}$ és el parell màxim que es vol aplicar i $\omega_p$ és la freqüència pròpia d'oscil·lació del parell d'entrada.

S'ha de tenir cura a l'hora de definir aquests dos paràmetres per motius diversos. Per començar, el parell màxim a aplicar varia en totes les articulacions i ha d'estar dintre dels límits de parell definits per a cadascuna d'elles. També cal vigilar que les freqüències d'oscil·lació que s'introdueixin no repercuteixin en cicles de parell massa llargs que puguin dur alguna articulació al seu límit articular i per tant fer malbé el robot. Per últim, cal tenir en compte que es compleixi el Teorema de Shannon (vegis l'Annex 1) en tot moment i que el nombre de períodes d'oscil·lació sigui raonablement gran. Com que el robot treballa amb un període de mostratge de $T = 0.02$s no serà cap problema complir la desigualtat de Shannon ja que treballarem amb cicles que tindran períodes d'oscil·lació de l'ordre de segons i per tant:

$$T \leq T_p/2, \quad T = 0.02\text{s}, \quad T_p \geq 0, \tag{6}$$

on $T$ és el període de mostratge del robot i $T_p = \dfrac{1}{\omega_p}$ és el període d'oscil·lació dels parells d'entrada.

De totes maneres, a més a més de complir el Teorema de Shannon, per cada període d'oscil·lació dels parells d'entrada el sistema haurà de prendre moltes mostres i es mirarà que el nombre de mostres per període d'oscil·lació $N$ sigui suficientment gran:

$$N = \frac{\omega}{\omega_p} = \frac{T_p}{T} >> 2 \tag{7}$$

De totes maneres, en el nostre cas, degut a la diferència dels ordres de magnitud entre els dos períodes, el nombre de mostres per període d'oscil·lació serà molt elevat i per tant no hi haurà cap pèrdua d'informació.

## 5.2 Experiments en llaç obert realitzats i anàlisi de les dades

Seguint la metodologia de treball per a la modelització de la velocitat explicada a l'apartat anterior, s'han anat executant d'una banda les trajectòries mitjançant parells sinusoidals a les diferents articulacions. A part, s'ha desconectat el controlador PID implementat de manera que s'han pogut controlar les variables d'entrada en tot moment per posteriorment observar el comportament del sistema davant d'aquestes entrades.

Pel que fa a la modelització, primer s'ha mirat de modelitzar la velocitat d'una articulació lliure, la 1, i posteriorment la d'una parella d'articulacions que compartís acoblament entre elles, la 2-3. Un cop modelitzades les velocitats, s'ha pogut estendre el procediment a la resta d'articulacions ja que el comportament de cada una d'elles serà similar a algun dels dos casos anteriors. Per a cada articulació s'han executat 10 trajectòries que ens han permès obtenir un volum de dades prou significatiu i que a la vegada podrà ser representatiu de la realitat dinàmica que experimenta el robot. Aquestes 10 trajectòries corresponen a cadascun dels paquets de dades per a la modelització dinàmica de cadascuna de les articulacions. Per a la validació dels models s'han executat 10 noves trajectòries i s'ha obtingut el segon paquet de dades. Per a validar els models s'han agafat les dades reals corresponents als parells aplicats i s'han comparat les velocitats reals del robot amb les que el model calculava per a cada trajectòria.

Un cop obtingudes les dades, aquestes s'han tractat amb el software de càlcul Matlab i utilitzant la llibreria d'identificació de sistemes s'han obtingut models prou bons, és a dir amb un ajust percentual de les dades d'un 75-80% com a mínim, per d'aproximar les velocitats a partir dels parells. Els models obtinguts i la seva estructura es detallaran a l'apartat següent.

Per tractar aquests conjunts de dades i obtenir un model de la forma (4) s'ha utilitzat la llibreria del Matlab anomenada *System Identification Toolbox* que permet obtenir molts models amb diferentes estructures i que permeten variacions en els seus paràmetres interns

que regulen el seu funcionament. Aquesta llibreria treballa un cop se li defineix un conjunt de dades que comporta la declaració de tots els *inputs* i els *outputs* que té el sistema juntament amb el període de mostratge amb el què s'han pres aquestes. Un cop definides les entrades i les sortides del sistema, utilitzant els menús de selecció de model i opcions pròpies de cada un d'ells s'han anat obtenint diferents models amb diferents nivell d'aproximació fins a trobar els més precisos i adients pel nostre cas.

De la mateixa manera, a la llibreria també se li poden introduir dades per simular models ja calculats i mirar si les validacions són correctes o no. A continuació es presenta el panell de treball de la llibreria *System Identification Toolbox*.

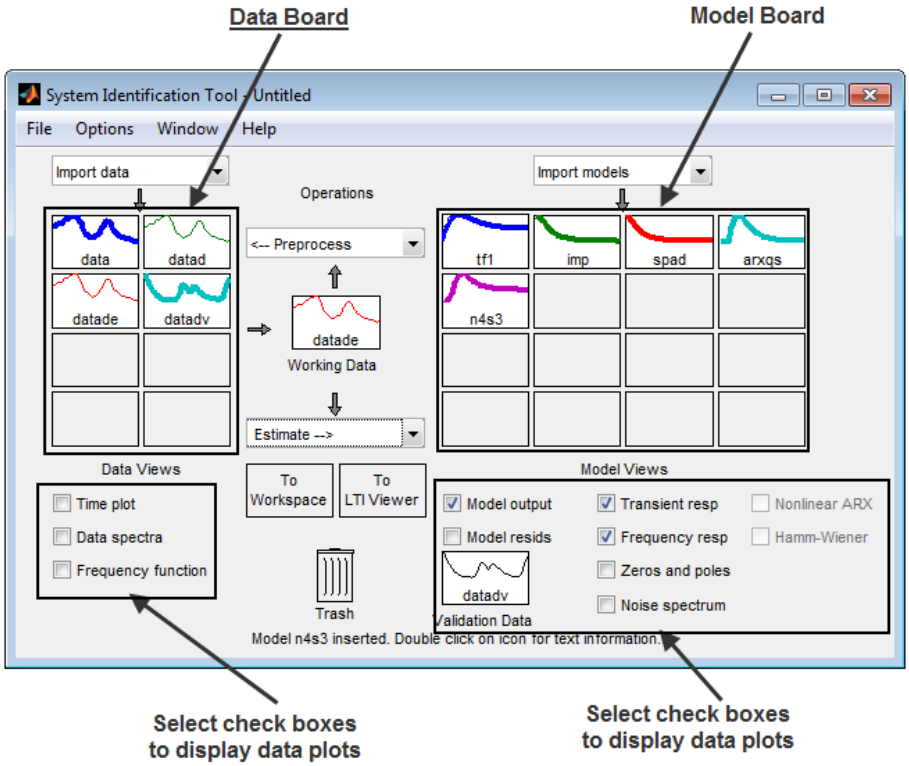


Figura 12: Panell de treball del *System Identification Toolbox*.

## 5.3 Indicadors per a avaluar els models

Per a saber si un model és bon aproximador o no, es necessitaran una sèrie d'indicadors que ens permetin avaluar el comportament del model a l'hora d'aproximar les dades que

es vulguin modelitzar.

Per defecte, el *System Identification Toolbox* dóna el valor del % d'aproximació (% fit) per a cada trajectòria que es vol aproximar amb cada model. A l'apartat següent, es parlarà dels valors dels percentatges que presenten els diversos tipus de models obtinguts que reflectiran la progressió i millora dels models nous respecte dels antics.

De cara a la validació dels models, s'ha volgut utilitzar el que s'anomenen KPIs (*Key Performance Indices*) que permeten fer-nos una idea de l'actuació del model a l'hora d'aproximar dades corresponents a diverses trajectòries. Aquests indicadors han estat per exemple l'error mitjà al quadrat MSE (*Mean Squared Error*) i l'error absolut mitjà MAE (*Mean Absolute Error*). Les fórmules per a calcular el MAE i el MSE són les següents:

$$MAE = \frac{1}{n}\sum_{r=1}^{n}\left|\hat{Y}_r - Y_r\right|$$

$$MSE = \frac{1}{n}\sum_{r=1}^{n}(\hat{Y}_r - Y_r)^2$$

on $n$ és el nombre de mostres amb les que es mostreja una trajectòria, $\hat{Y}_r$ és el valor estimat pel model corresponent a la mostra $r$-èssima, i $Y_r$ és el valor real de la mostra en el mateix instant. En el nostre cas, $\hat{Y}_r$ i $Y_r$ corresponen a les velocitats modelitzades i reals de cada articulació a cada instant.

Aquests indicadors es podran utilitzar per comparar models i decidir quin d'ells és més bon aproximador.

## 5.4 Descripció dels models obtinguts

Abans de parlar dels models introduïm la notació necessària per entendre les equacions que es veuran a continuació: anomenem $u(t)$ al conjunt d' *inputs* que tindran els nostres diferents models que correspondran als parells aplicats per a cada conjunt d'articulacions, i $y(t)$ al conjunt d' *outputs* que tindran els diferents models i que correspondran a les velocitats de cada articulació del conjunt. Es poden descriure de la següent manera:

$$\begin{cases} u_i(t) = [\tau_i(t)]^T, & i = 1, 4, 7 \\ u_i(t) = [\tau_2(t), \tau_3(t)]^T, & i = 2, 3 \\ u_i(t) = [\tau_5(t), \tau_6(t)]^T, & i = 5, 6 \\ y_i(t) = \dot{\theta}_i(t), & i = 1:7. \end{cases}$$

Les equacions que es presentaran a continuació per a definir cadascun dels models estan definides com si el model només tingués una entrada i una sortida. Per les articulacions en que es necessita més d'una entrada s'ha utilitzat el modelat MIMO (*Multiple Input Multiple Ouput*).

Es presenten tot seguit, els diferents models amb els que es va treballar per aproximar les diferentes velocitats.

Els primers tipus van ser models lineals polinòmics [17], que presenten l'estructura següent:

$$A_i(q)y_i(t) = \frac{B_i(q)}{F_i(q)}u_i(t) + \frac{C_i(q)}{D_i(q)}e_i(t)$$

on $A_i(q)$, $B_i(q)$, $C_i(q)$, $D_i(q)$ i $F_i(q)$ són polinomis de diferents graus, a especificar, en funció de l'*operador retard* $q$. Aquest operador s'anomena operador retard perquè el que fa és obtenir valors de les variables en instants passats (la llibreria permet triar el nombre d'instants passats als quals recórrer). Aquest fet permetrà estimar la velocitat actual de qualsevol articulació a partir no només dels parells actuals sinó també dels parells i velocitats d'instants passats en el temps.

De tota aquesta família de models els primers que es van mirar d'obtenir van ser els models ARX que presenten una estructura una mica més simple en comparació amb els models lineals polinòmics generals i per tant més fàcils de ser implementats en codi. Els models ARX es poden representar per:

$$A_i(q)y_i(t) = \sum_{m=1}^{nu} B_{i_m}(q)u_i(t - nk_m) + e_i(t) \tag{8}$$

on $B_{i_m}$ són les components de $B_i(q)$.

També es van provar la resta d'estructures per als models lineals com els models ARMAX, els models d'Espai d'Estats o els models de Box-Jenkins. Les equacions que defineixen aquests models són les següents:

Per als models ARMAX:

$$A_i(q)y_i(t) = \sum_{m=1}^{nu} B_{i_m}(q)u_i(t - nk_m) + C_i(q)e_i(t). \tag{9}$$

Per als models d'Espai d'Estats:

$$\begin{cases} \dot{x}_i(t) = Fx_i(t) + G \cdot u_i(t), \\ y_i(t) = Hx_i(t), \\ x_i(0) = x0. \end{cases} \tag{10}$$

on $x_i(t)$ és l'estat de l'articulació $i$ a l'instant $t$.

Pels models Box-Jenkins:

$$A_i(q)y_i(t) = \frac{B_i(q)}{F_i(q)}u_i(t - nk_m) + \frac{C_i(q)}{D_i(q)}e_i(t) \tag{11}$$

Quan es van provar els tres mètodes els resultats obtinguts van ser similars.

A l'hora de mirar d'obtenir els millors models possibles amb alguna d'aquestes estructures, el *System Identification Toolbox* ofereix la possibilitat de modificar paràmetres propis dels models com ara els diferents graus dels polinomis que operaven amb l'operador de retard $q$ (en els models ARX, ARMAX i Box-Jenkins) o el nombre d'estats i l'estructura de les matrius amb les que treballen els models d'Espais d'Estats.

Els models lineals polinòmics no van donar resultats suficientment bons, com ja era d'esperar des d'un principi, ja que intentàvem aproximar un fenòmen clarament no lineal amb funcions lineals.

Es va provar aleshores d'obtenir directament models no lineals. Tractant acuradament les no linealitats, es van trobar els models no lineals amb l'estructura de Hammerstein-Wiener [16]. Aquests models consten de tres blocs que permeten transformar els parells d'entrada en les corresponents velocitats de sortida. Tenen l'estructura que es mostra a la Figura 13:

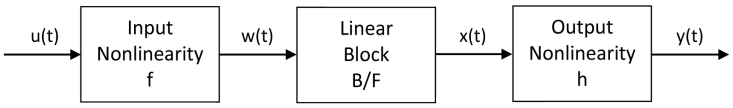


Figura 13: Estructura d'un model Hammerstein-Wiener.

El primer bloc correspon a una funció lineal a trossos on es pot definir el nombre de trossos que es vol que presenti juntament amb la localització (coordenada $x$ o coordenades $x$ i $y$) dels punts que delimiten els trams lineals de cada tros. La sortida d'aquest primer bloc va a parar a un segon bloc que correspon a un sistema lineal on tenim la llibertat de triar el nombre de zeros i de pols que es desitja que aquest tingui. Per últim, a la sortida del sistema lineal tenim un tercer bloc que correspon a una nova funció lineal a trossos que presenta les mateixes opcions que el primer bloc pel que fa a la definició dels seus paràmetres.

El comportament dels tres blocs es pot observar a les Figures 14, 15 i 16:

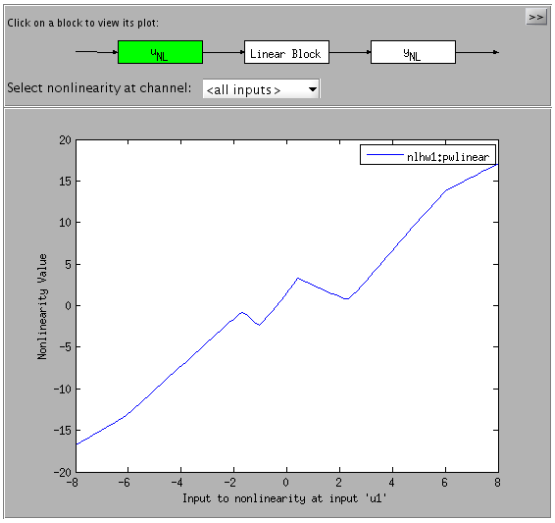


Figura 14: Funció lineal a trossos del primer bloc.

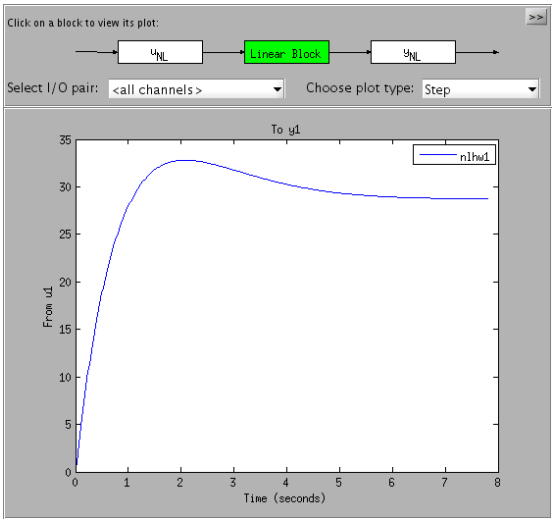


Figura 15: Resposta del sistema lineal de tercer ordre davant entrada graó.

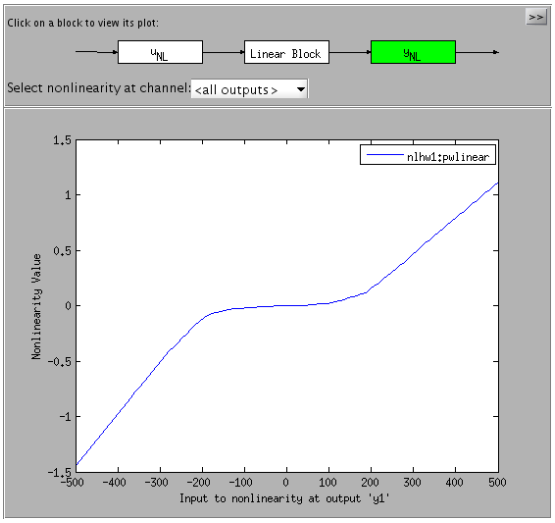


Figura 16: Funció lineal a trossos del tercer bloc.

Aquests diferents blocs es poden definir mitjançant equacions on $u_i(t) = \tau_i(t)$, $y_i(t) = \dot{\theta}_i(t)$ i $X_i(t)$ com l'estat del sistema lineal a l'instant $t$, on $i$ és el subíndex que indica a quina articulació ens estem referint, de la manera següent:

Primer bloc:

$$w_i(t) = w_{i_j} + \frac{(u_i(t) - u_{i_j})(w_{i_{j+1}} - w_{i_j})}{(u_{i_{j+1}} - u_{i_j})},$$

per algun $j$.

Segon bloc:

$$\begin{cases} \dot{X}_i(t) &= FX_i(t) + Gw_i(t), \\ x_i(t) &= HX_i(t), \\ X_i(0) &= X0, \end{cases}$$

Tercer bloc:

$$y_i(t) = y_{i_k} + \frac{(x_i(t) - x_{i_k})(y_{i_{k+1}} - y_{i_k})}{(x_{i_{k+1}} - x_{i_k})},$$

per alguna $k$.

on totes les parelles de punts $(u_{i_j}, w_{i_j})$ i $(x_{i_k}, y_{i_k})$ corresponen als punts que delimiten els trams lineals de les funcions a trossos del primer i tercer bloc per l'articulació $i$.

Les equacions del bloc lineal estan escrites en la forma d'Espai d'Estats ja que és molt més còmode i fàcil d'implementar en qualsevol codi. Aquestes equacions es poden reescriure en temps discret mitjançant les següents expressions:

Primer bloc:

$$w_i(KT) = w_{i_j} + \frac{(u_i(KT) - u_{i_j})(w_{i_{j+1}} - w_{i_j})}{(u_{i_{j+1}} - u_{i_j})}, \tag{12}$$

per algun $j$.

Segon bloc:

$$\begin{cases} X_i(KT+T) = AX_i(KT) + Bw_i(KT), \\ x_i(KT) = CX_i(KT), \\ X_i(0) = X0, \end{cases} \tag{13}$$

Tercer bloc

$$y_i(KT) = y_{i_k} + \frac{(x_i(KT) - x_{i_k})(y_{i_{k+1}} - y_{i_k})}{(x_{i_{k+1}} - x_{i_k})}, \tag{14}$$

per algun $k$.

Amb aquests models es va obtenir un % d'aproximació que oscil·lava entre el 85% i el 95%. Es va considerar aquests percentatges com a suficientment bons per al nostre sistema.

Tot i que els models amb estructura Hammerstein-Wiener eren molt precisos a l'hora d'aproximar les velocitats a partir dels parells aplicats, es van haver de rebutjar perquè els models a utilitzar per tal que el controlador calculés els parells a aplicar en tot moment havien de permetre una realimentació senzilla del sistema. Aquesta realimentació no es presenta amb els models de Hammerstein-Wiener degut a que tant els estats com les variables internes entre blocs no tenen interpretació física evident i això fa que no se'ls pugui relacionar amb la velocitat.

Es va buscar aleshores un model que presentés una estructura també no lineal però que permetés millor la realimentació en llaç tancat. Aquest va ser el cas dels models ARX no lineals [15]. Aquests models són una extensió dels models lineals ARX i presenten un bloc no lineal en paral·lel amb el bloc lineal. Aquesta estructura es pot observar a la Figura 17 següent:

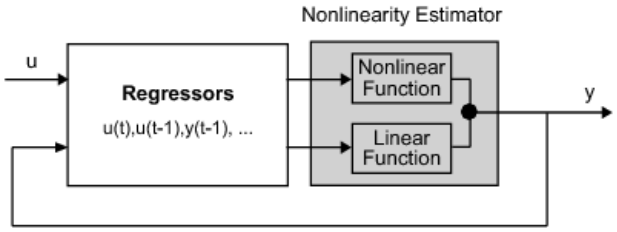


Figura 17: Estructura d'un model ARX no lineal.

L'expressió genèrica de les entrades del sistema amb les sortides és de la següent forma:

$$\begin{cases} y(t) = F(x(t)) = L^T(x(t) - r) + d + g(Q(x(t) - r)) \\ x(t) = [y(t-1), y(t-1), ..., y(t-t_y), u(t-1), u(t-2), ..., u(t-t_u)] \end{cases}$$

on $L^T(x - r) + d$ és la sortida del bloc lineal i $g(Q(x - r))$ la sortida del bloc no lineal. La variable $x(t)$ fa referència al conjunt de regressors utilitzats pels diferents blocs i $r$ és la mitjana dels regressors mesurada durant la modelització. Finalment, $t_y$ i $t_u$ són els instants passats que es volen consultar, tant de les sortides com de les entrades.

Com es pot observar, aquests models presenten tres blocs que es comentaran breument tot seguit.

D'una banda, es pot observar un bloc que té com a entrades els *inputs* $u(t)$ del sistema i les sortides $y(t)$ del sistema realimentades i que permet triar quins seran els regressors amb els que treballarà el model. Aquest fet, permet seleccionar quines entrades i sortides del sistema es volen utilitzar per calcular i fins a quin instant passat en el temps es vol consultar. Triant adequadament aquests regressors s'obtindran models suficientment bons.

Per cadascuna de les articulacions del robot el regressors utilitzats han estat els següents:

$$\begin{cases} x_i(KT) = [\theta_i(KT-T), \theta_i(KT-2T), \tau_i(KT-T), \tau_i(KT-2T)], \quad i = 1,4,7 \\ x_i(KT) = [\theta_i(KT-T), \theta_i(KT-2T), \tau_2(KT-T), \tau_2(KT-2T), \tau_3(KT-T), \\ \qquad \tau_3(KT-2T)], \quad i = 2,3 \\ x_i(KT) = [\theta_i(KT-T), \theta_i(KT-2T), \tau_5(KT-T), \tau_5(KT-2T), \tau_6(KT-T), \\ \qquad \tau_6(KT-2T)], \quad i = 5,6 \end{cases}$$

A continuació, aquests regressors són utilitzats per dos blocs que treballen en paral·lel (un de lineal i un de no lineal) la sortida dels quals sumada dóna el valor de sortida del model. El bloc lineal no té cap opció a definir ja que mitjançant els regressors ja estan determinats els graus dels polinomis que operen amb l'operador $q$. En canvi, el programa dóna certa llibertat de tria respecte de les funcions amb les que opera el bloc no lineal. Entre aquestes funcions hi ha la *wavenet network*, la *sigmoid network* i la *tree partition*. La *wavenet network* no va donar massa bons resultats i es va descartar des d'un principi. Entre les altres dues els resultats van ser similars però es va acabar seleccionant la *sigmoid network* degut a que era molt més fàcil d'implementar en codi. Aquesta funció *sigmoid* és una funció matemàtica utilitzada bastant en temes de probabilitat i logística que té l'expressió i la forma següents:

$$S(t) = \frac{1}{1 + e^{-t}}$$

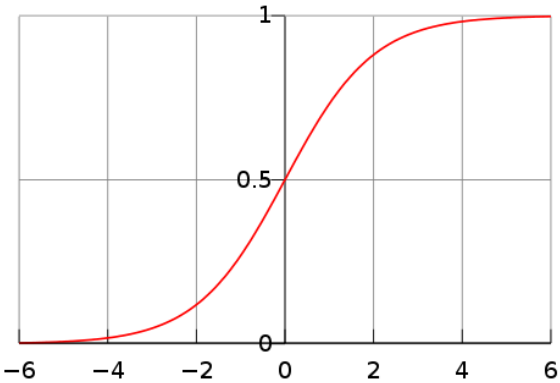


Figura 18: Funció *Sigmoid* en funció del temps

Substituint aquesta funció *sigmoid* a l'expressió (15) s'obté l'expressió particular que tindrà el nostre model:

$$y(t) = F(x(t)) = (x(t) - r)PL + \sum_{k=1}^{n} a_k f((x(t) - r)Qb_k + c_k) + d$$

on

$$f(t) = \frac{1}{1 + e^{-t}}$$

Aquestes equacions reescrites convenientment en temps discret tenen les següents expressions:

$$y(KT + T) = F(x(KT)) = (x(KT) - r)PL + \sum_{k=1}^{n} a_k f((x(KT) - r)Qb_k) + d \qquad (15)$$

on

$$f(KT) = \frac{1}{1 + e^{-KT}}$$

## 5.5 Validació dels models

En aquesta secció es mostraran els diferents nivells d'aproximació que han donat tots els models obtinguts. Els models es mostraran en l'ordre cronològic en què s'han obtingut. Els models es compararan tant gràficament com mitjançant els diferents indicadors que evaluen els models explicats a l'apartat 5.3. Tots els resultats es presentaran només per l'articulació 1 ja que per les altres articulacions els resultats han estat similars.

Com ja s'ha comentat a l'apartat anterior, els primers models que es van obtenir van ser els models polinòmics, encara que no van donar resultats prou satisfactoris. A continuació es presenten un conjunt d'imatges corresponents a diverses aproximacions obtingudes amb aquests models polinòmics, concretament amb els ARX. Les imatges de l'esquerra corresponen a les velocitats desitjades en dues trajectòries i les corresponents aproximacions que el model dóna, i les imatges de la dreta corresponen a les posicions desitjades per aquestes trajectòries comparades amb les posicions que s'obtenen en integrar les velocitats modelitzades:

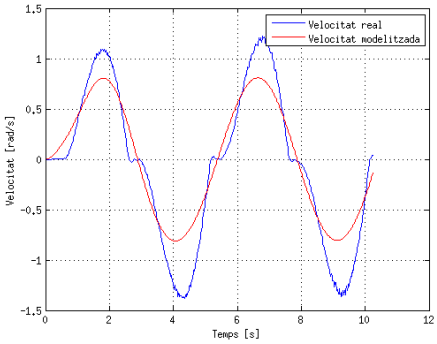


Figura 19: Velocitats desitjades i modelades amb el model ARX per a la trajectòria A segons l'equació (8).

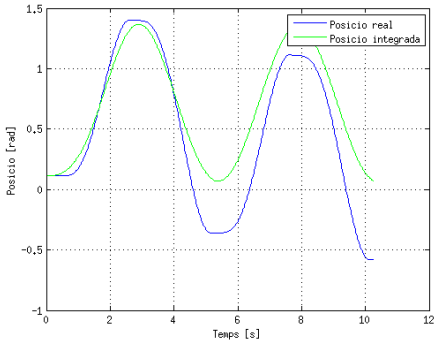


Figura 20: Posicions desitjades i integrades per a la trajectòria A.

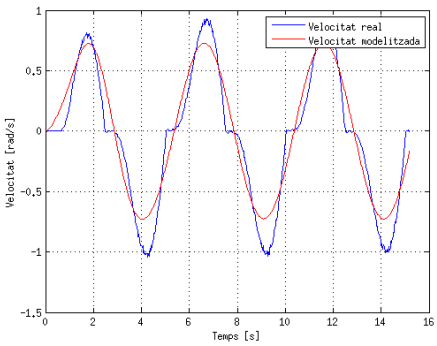


Figura 21: Velocitats desitjades i modelades amb el model ARX per a la trajectòria B segons l'equació (8).

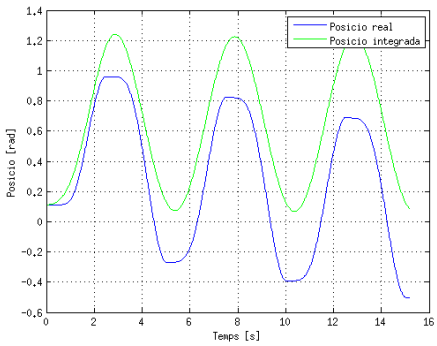


Figura 22: Posicions desitjades i integrades per a la trajectòria B.

Com es pot observar a les Figures 19 i 21, l'aproximació de la velocitat no és suficientment bona i per tant la seva integració condueix a una trajectòria que dista de la realitat (vegi's les Figures 20 i 22). El % d'aproximació de la velocitat per part del model oscil·lava entre el 50 i el 75% , per la cual cosa ja es va preveure que les trajectòries no serien les desitjades.

També es van provar altres estructures per als models lineals però els resultats no va ser gaire millors respecte dels obtinguts amb els models ARX. A continuació es presenten gràficament els resultats obtinguts pel model ARMAX, pel model d'espai d'estats i pel model BOX-JENKINS per a dues trajectòries del conjunt de validació:

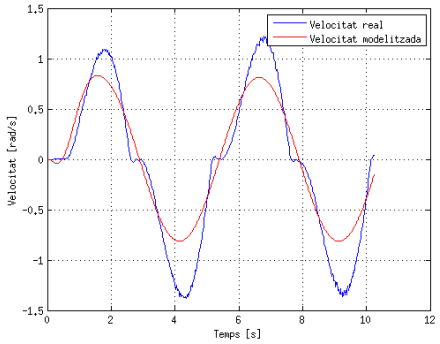


Figura 23: Velocitats desitjades i modelades amb el model ARMAX per a la trajectòria A segons l'equació (9).

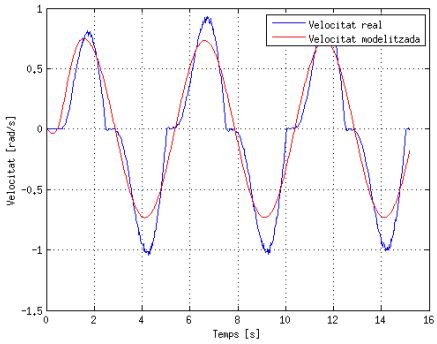


Figura 24: Velocitats desitjades i modelades amb el model ARMAX per a la trajectòria B segons l'equació (9).

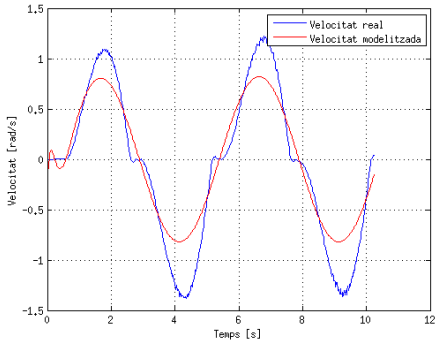


Figura 25: Velocitats desitjades i modelades amb el model d'espai d'estats per a la trajectòria A segons l'equació (10).

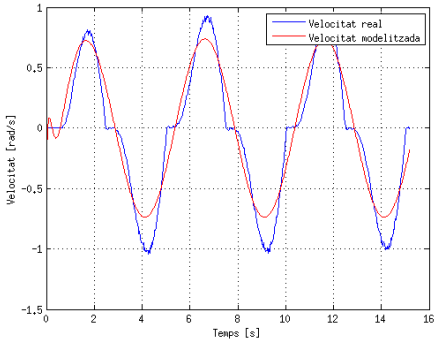


Figura 26: Velocitats desitjades i modelades amb el model d'espai d'estats per a la trajectòria B segons l'equació (10).

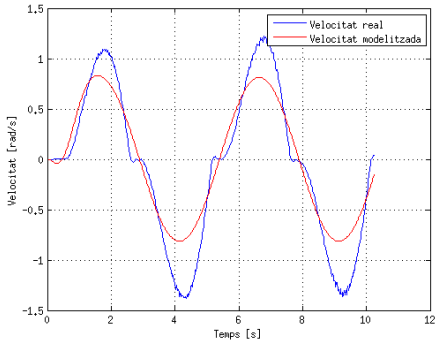


Figura 27: Velocitats desitjades i modelades amb el model Box-Jenkins per a la trajectòria A segons l'equació (11).

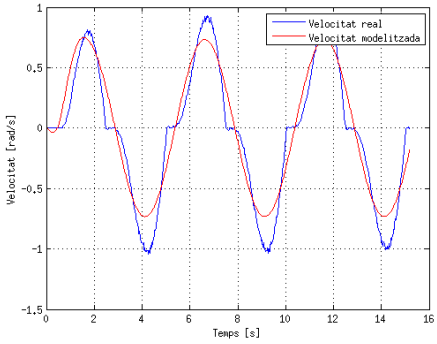


Figura 28: Velocitats desitjades i modelades amb el model Box-Jenkins per a la trajectòria B segons l'equació (11).

Com es pot observar a les Figures 23, 24, 25, 26, 27 i 28, l'aproximació no millora suficientment. Per millorar els resultats obtinguts amb els models lineals es va mirar d'obtenir models no lineals més complexos que fossin capaços de tractar les no linealitats del sistema convenientment. Els primers models que es van provar van ser els que presentaven l'estructura Hammerstein-Wiener que tenien 3 blocs (2 blocs de funcions lineals a trossos i 1 bloc lineal al mig) i com s'esperava, es va observar un salt qualitatiu en els resultats.

Seguint el procediment fet servir pels models lineals, es presenten gràficament els resultats obtinguts per a dues de les trajectòries per validar el model Hammerstein-Wiener:

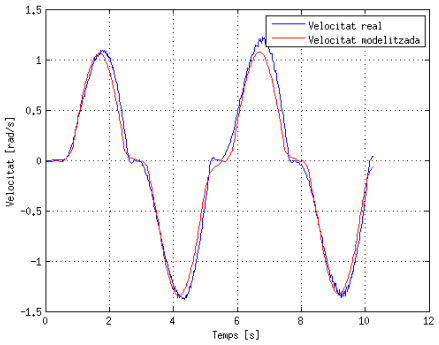


Figura 29: Comparació entre la velocitat real i la modelada amb el model HW per a la trajectòria A segons les equacions (12), (13) i (14).

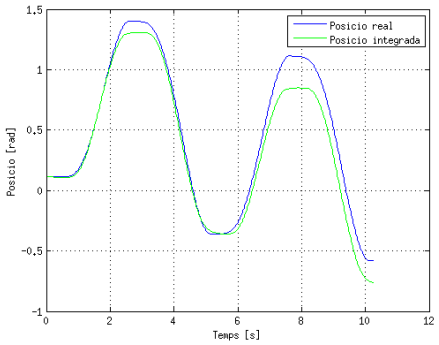


Figura 30: Comparació entre la posició real i la integrada per a la trajectòria A.

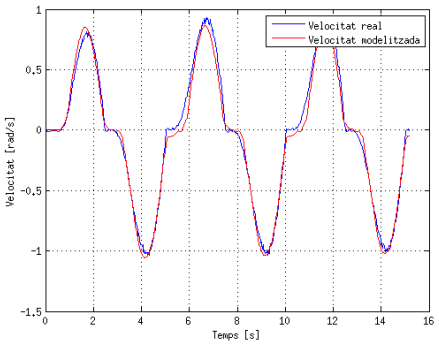


Figura 31: Comparació entre la velocitat real i la modelada amb el model HW per a la trajectòria B segons les equacions (12), (13) i (14).

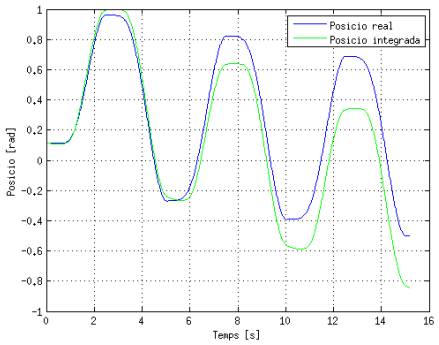


Figura 32: Comparació entre la posició real i la integrada per a la trajectòria B.

S'observa a les Figures 29 i 31 que el nivell d'aproximació de la velocitat és molt notable i per tant la validació es pot considerar correcta. Malgrat això, es pot observar que la posició integrada (vegi's les Figures 30 i 32), a mesura que avança el temps, dista cada cop més de la realitat, però cal tenir en compte que això no és més que un experiment en llaç obert i que podrem corregir aquestes desviacions quan fem el llaç tancat.

Malauradament, aquests models no es van poder utilitzar com a models de predicció en el control predictiu ja que són molt difícils de realimentar. Per això, es va mirar de trobar altres models no lineals que donessin nivells d'aproximació, com a mínim similars. Aquests nous models van ser els models ARX no lineals. Els resultats obtinguts es poden representar a continuació mitjançant les figures que corresponen a dues de les trajectòries:

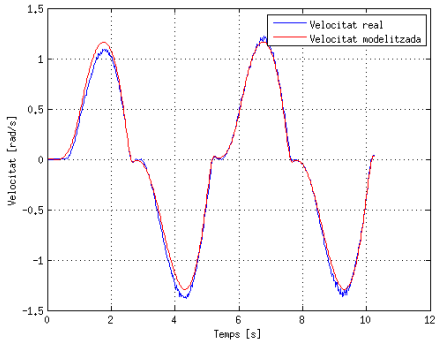


Figura 33: Comparació entre la velocitat real i la modelada amb el model ARX no lineal per a la trajectòria A segons l'equació (15).

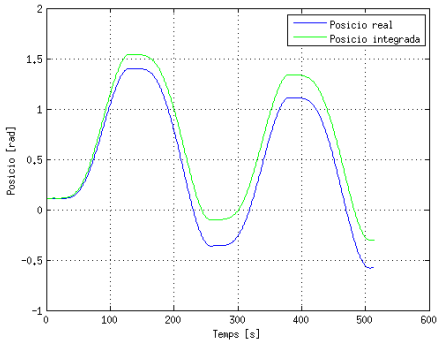


Figura 34: Comparació entre la posició real i la integrada per a la trajectòria A.

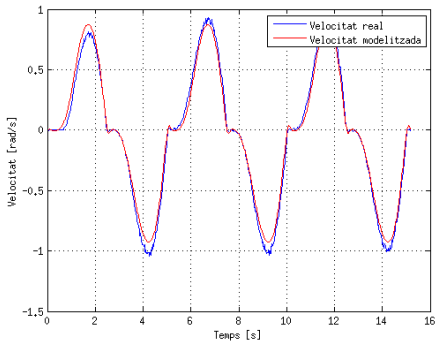


Figura 35: Comparació entre la velocitat real i la modelada amb el model ARX no lineal per a la trajectòria B segons l'equació (15).

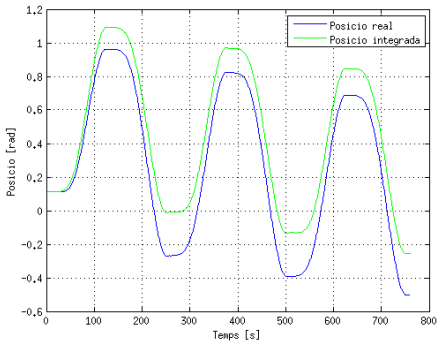


Figura 36: Comparació entre la posició real i la integrada per a la trajectòria B.

Com es pot observar a les figures 33 i 35, el nivell d'aproximació és bastant similar a l'obtingut amb els models no lineals Hammerstein-Wiener.

A més a més d'una comparació gràfica, també s'ha dut a terme una comparació entre els models polinòmics, els models no lineals Hammerstein-Wiener i els models ARX no lineals fent servir els KPIs i el percentage d'aproximació per a cada trajectòria i model. Els resultats es presenten a les tres taules següents:

| Trajectòries | MAE ARX | MAE ARMAX | MAE SS | MAE BJ | MAE HW | MAE NARX |
|---|---|---|---|---|---|---|
| Trajectòria A | 0.2146 | 0.2069 | 0.2050 | 0.2069 | **0.0750** | 0.0911 |
| Trajectòria B | 0.1412 | 0.1387 | 0.1359 | 0.1387 | 0.0715 | **0.0588** |
| Trajectòria C | 0.1488 | 0.1506 | 0.1502 | 0.1506 | 0.0583 | **0.0506** |
| Trajectòria D | 0.1390 | 0.1280 | 0.1272 | 0.1280 | **0.0509** | 0.0613 |
| Trajectòria E | 0.1377 | 0.1268 | 0.1304 | 0.1268 | **0.0417** | 0.0495 |
| Trajectòria F | 0.1297 | 0.1212 | 0.1330 | 0.1212 | 0.0526 | **0.0434** |
| Trajectòria G | 0.1456 | 0.1489 | 0.1553 | 0.1489 | **0.0387** | 0.0513 |
| Trajectòria H | 0.1621 | 0.1621 | 0.1698 | 0.1621 | **0.0458** | 0.0523 |
| Trajectòria I | 0.1553 | 0.1558 | 0.1543 | 0.1558 | **0.0613** | 0.0763 |
| Trajectòria J | 0.1666 | 0.1661 | 0.1643 | 0.1661 | 0.0937 | **0.0922** |

| Trajectòries | MSE ARX | MSE ARMAX | MSE SS | MSE BJ | MSE HW | MSE NARX |
|---|---|---|---|---|---|---|
| Trajectòria A | 0.0714 | 0.0675 | 0.0660 | 0.0675 | **0.0115** | 0.0181 |
| Trajectòria B | 0.0273 | 0.0266 | 0.0252 | 0.0266 | 0.0095 | **0.0072** |
| Trajectòria C | 0.0316 | 0.0330 | 0.0320 | 0.0330 | 0.0067 | **0.0054** |
| Trajectòria D | 0.0283 | 0.0247 | 0.0245 | 0.0247 | **0.0052** | 0.0092 |
| Trajectòria E | 0.0283 | 0.0235 | 0.0243 | 0.0235 | **0.0037** | 0.0056 |
| Trajectòria F | 0.0290 | 0.0252 | 0.0284 | 0.0252 | 0.0051 | **0.0038** |
| Trajectòria G | 0.0311 | 0.0326 | 0.0344 | 0.0326 | **0.0035** | 0.0079 |
| Trajectòria H | 0.0353 | 0.0354 | 0.0378 | 0.0354 | **0.0049** | 0.0063 |
| Trajectòria I | 0.0332 | 0.0344 | 0.0329 | 0.0344 | **0.0057** | 0.0120 |
| Trajectòria J | 0.0392 | 0.0391 | 0.0377 | 0.0391 | **0.0176** | 0.0200 |

| Trajectòries | %fit ARX | %fit ARMAX | %fit SS | %fit BJ | %fit HW | %fit NARX |
|---|---|---|---|---|---|---|
| Trajectòria A | 65.65 | 66.07 | 66.36 | 67.24 | 86.79 | **93.27** |
| Trajectòria B | 71.00 | 71.22 | 71.93 | 72.58 | 89.53 | **90.18** |
| Trajectòria C | 49.88 | 48.91 | 49.85 | 49.99 | 85.85 | **88.45** |
| Trajectòria D | 75.62 | 76.28 | 76.33 | 76.37 | 88.17 | **93.21** |
| Trajectòria E | 76.73 | 77.15 | 77.15 | 77.57 | 87.42 | **91.15** |
| Trajectòria F | 64.55 | 65.80 | 64.79 | 66.10 | 83.69 | **93.97** |
| Trajectòria G | 49.85 | 48.07 | 48.96 | 50.10 | 82.81 | **84.55** |
| Trajectòria H | 67.01 | 65.39 | 65.46 | 66.04 | **89.57** | 85.16 |
| Trajectòria I | 60.13 | 57.03 | 58.04 | 58.31 | **87.57** | 86.60 |
| Trajectòria J | 67.43 | 66.57 | 67.14 | 69.20 | 89.72 | **90.57** |

Observant aquestes taules es pot concloure que la capacitat dels dos models no lineals a l'hora d'aproximar és molt similar i, per tant, el nivell d'aproximació obtingut amb els models ARX no lineals serà suficientment bo.

Un cop obtinguts els models definitius es passarà ara a parlar de com s'ha controlat el sistema i quines han estat les tècniques de control que s'han fet servir.

# 6 Control del sistema basat en optimització

## 6.1 Idea de control

Un cop obtingut un model prou bo es va pensar en quina estratègia de control seria més adient seguir. Les tècniques de control que fins llavors s'havien implementat al WAM corresponien a tècniques de control clàssiques en les que es tracta normalment amb sistemes lineals invariants en el temps on les no linealitats requereixen tractaments especials. Aquestes tècniques servien per compensar els errors de posició del robot al llarg de les seves trajectòries i s'aconseguien mitjançant controladors PIDs. Encara que aquestes tècniques continuen sent vàlides i donant bons resultats, vam voler pensar en noves tècniques de control que donessin resultats millors.

Des d'un principi es va pensar en una tècnica de control que s'anomena *control predictiu basat en models*. El control predictiu, que pertany a la família dels controladors multivariables, és una estrategia de control que s'ha imposat molt en les últimes dècades a la indústria de processos com la millor opció per controlar un procés amb múltiples entrades i sortides, satisfent les restriccions d'operació del propi sistema. Aquesta estratègia de control dissenya algoritmes de control de fàcil implementació i que facin ús de models lineals o no lineals per a una operació òptima del procés en tot el seu rang de funcionament.

El control predictiu haurà de permetre garantir un seguiment precís de les trajectòries mitjançant un programa d'optimització que calculi els parells necessaris a aplicar sobre les articulacions del robot en tot instant. Aquest programa d'optimització utilitzarà la funció `fmincon` que permet minimitzar una funció de cost tot complint les restriccions del sistema.

D'aquesta manera, quan introduïm una sèrie de parells a les articulacions en diferents instants podrem saber a quina velocitat s'anirà movent el robot i per tant per quines posicions anirà passant (sempre que coneixem la posició inicial del robot abans d'aplicar els parells). Aquest model s'anomenarà *model de simulació*. Anàlogament, si nosaltres obtenim un model que aproxima prou bé la dinàmica del robot i volem seguir una trajectòria en la que coneixem les posicions i les velocitats desitjades en tot moment, hauríem de ser capaços d'utilitzar el model per saber quins parells aplicar per seguir les

trajectòries desitjades de la millor manera possible. Aquest model serà el que anomenarem *model de predicció*. Per a realitzar aquesta predicció caldrà que el controlador que contingui el model de predicció disposi també de la informació corresponent a les posicions i velocitats desitjades que vol seguir. Aquesta informació addicional només s'utilitzarà per dur a terme comparacions amb els termes reals que surtin del robot.

Aquest model de predicció és el que ens ha de servir per aconseguir un bon seguiment de les trajectòries del robot. Per fer-ho utilitzarem una llibreria del MATLAB anomenada *Optimization Toolbox* i més en concret una funció d'aquesta llibreria que s'anomena `fmincon` que minimitza una funció subjecta a unes restriccions. Per això, es plantejarà una funció objectiu (o també anomenada *de cost*) en funció de les variables amb les que el sistema opera i que estaran sotmeses a una sèrie de restriccions. En el nostre cas la funció objectiu serà minimitzar l'error de posició en les trajectòries, les variables del sistema seran els parells a les articulacions i les restriccions seran les restriccions de parell, de posició i de velocitat.

El model servirà per relacionar els parells d'entrada amb les velocitats i les posicions del robot. Per tant, caldrà implementar correctament en codi tant el model, com la funció objectiu i les restriccions, que en el nostre cas seran desigualtats no lineals. En tot això juga un paper important el fet que la funció `fmincon` pugui treballar amb restriccions que siguin desigualtats no lineals com és el cas del nostre model. Tot això es pot escriure en equacions genèriques de la següent manera:

$$\min_x(f(x))$$

restringida a

$$\begin{cases} c(x) & \leq & 0 \\ c_{eq}(x) & = & 0 \\ Ax & \leq & b \\ A_{eq}x & = & b_{eq} \\ lb \leq & x & \leq ub \end{cases}$$

on $f(x)$ és la funció objectiu (error de posició de les trajectòries al llarg del temps), $x$ és la variable a manipular (el parells en les articulacions), $c(x)$ i $ceq(x)$ representen les desigualtats i les igualtats no lineals del sistema (en el nostre model només hi ha desigualtats), les matrius $A$ i $A_{eq}$ i els vectors $b$ i $b_{eq}$ són constants (en el nostre model són nuls ja que el sistema no presenta restriccions lineals) i $lb$ i $ub$ són els valors mínims i màxims que les variables a controlar poden prendre en qualsevol moment.

Els codis corresponents a aquest problema d'optimització, on es definiran per separat les funcions $f(x)$ i $c(x)$ (vegi's l'annex 12.4), s'escriuran fent servir el MATLAB que és on s'executarà la funció `fmincon`.

Gràcies a aquest programa es podrà dur a terme la predicció dels parells que cal aplicar per seguir una trajectòria desitjada. El model de predicció serà, per tant, el que utilitzarà el controlador per decidir quins parells aplicar en tot moment. Per altra banda, el model de simulació s'utilitzarà per simular com reacciona el robot davant d'aquestes entrades de parell. Sovint es fa servir el mateix model per a la predicció i per a la simulació però, per a la simulació convé utilitzar el model que millor aproximi la realitat ja que el robot no es comporta exactament com diu el model de predicció i les desviacions es poden corregir realimentant el sistema. Es pot pensar que si s'utilitza un model per fer prediccions i després un model diferent per simular la planta els resultats no coincidiran, però gràcies a la realimentació del sistema aquests efectes es poden corregir. Sempre es pot pensar que el robot no es comporta al 100% com el model de predicció i per tant el més intel·ligent serà utilitzar com a model de simulació el model que millor modelitzi aquest comportament.

A l'apartat següent s'explicaran quins models de predicció i simulació s'han utilitzat juntament amb la forma de realimentar el sistema.

## 6.2 Model de predicció, model de simulació i realimentació del sistema

Tal com s'ha dit a l'apartat anterior, en el nostre problema es treballarà amb dos models. Un d'ells s'utilitzarà per fer prediccions respecte els parells a aplicar per seguir certes trajectòries i l'altre s'utilitzarà per simular el comportament del robot davant els parells que s'hagin predit. Fins ara, només s'havia parlat d'un model que aproximés el millor

possible el comportament de la planta del nostre sistema (el robot WAM) per després fer proves de simulació utilitzant aquest model com a planta; però quan es va decidir utilitzar el control predictiu com a estratègia de control, es va necessitar un segon model. En el control predictiu el controlador utilitza un model que aproxima el comportament del robot per a fer les respectives prediccions que posterioment entrega a la planta. A partir del model de predicció, el controlador ha de ser capaç en tot instant de determinar el parell a aplicar per seguir les trajectòries i posteriomente entregar-lo com a entrada de la planta. El cas ideal seria utilitzar aquestes entrades predites i executar-les aplicant-les directament al robot WAM per veure la resposta d'aquest, però això implicaria la implementació de tots els programes de control i d'optimització i el model fets en aquest projecte en llenguatge de codi C++. Aquesta implementació no es contemplarà dins del marc d'aquest treball donat que la seva complexitat requeria molt de temps. Per arreglar aquest problema, s'utilitzarà un model de simulació que s'explicarà més endavant i que reprodueixi el comportament de la planta i així s'obtindran les sortides del sistema. Com que el model no es comporta al 100% com el robot, sempre hi haurà diferències entre la realitat i les simulacions que seran més petites com més bo sigui el model. Es per tant essencial obtenir bons models aproximadors. Com més bon aproximador sigui el model més precises i correctes seran les prediccions a fer de cara als parells a aplicar i més realista serà la simulació del comportament de la planta real.

De totes maneres no tots els models es poden utilitzar com a models de predicció. En efecte, el model de predicció està contingut dins del controlador i per tant ha de permetre la realimentació correcta del sistema, que en el nostre cas és de velocitat. Això fa, per exemple, que els models Hammerstein-Wiener no s'hagin triat com a models de predicció ja que per causa de l'estructura en blocs del model i pel fet que tant els estats del sistema lineal com les variables internes del model no tenen interpretació física, no es pot relacionar les variables internes directament amb la velocitat. En canvi, els models ARX no lineals utilitzen les dades de parell i velocitats en instants anteriors per formar el regressor que permet calcular la velocitat en cada instant. La realimentació consisteix aleshores en anar actualitzant aquest regressor afegint en cada instant els parells i velocitats nous. D'aquesta manera, un cop simulat el comportament de la planta davant d'una entrada s'agafarà la dada corresponent a la velocitat a l'instant actual i s'enviarà directament al controlador

per actualitzar el regressor. Com a models de predicció s'han utilitzat, doncs, els models ARX no lineals.

A l'hora de triar models de simulació no hi ha hagut tants problemes ja que només havien de ser capaços de calcular les velocitats a partir dels parells i velocitats anteriors. En aquest sentit, per a simular el comportament de la planta es poden utilitzar tant els models no lineals Hammerstein-Wiener com els models ARX no lineals. Com que s'ha observat que en moltes articulacions els models Hammerstein-Wiener i els models ARX no lineals aproximaven de manera similar, s'ha decidit realitzar proves utilitzant els dos models diferents de simulació i després observar quin dóna millors resultats.

## 6.3 Optimització amb el `fmincon`

En primer lloc, les equacions que defineixen el problema d'optimització reescrites per al nostre cas concret són les següents:

$$min_{\tau_i(t)}(\theta_{real_i}(t) - \theta_i(t))^2$$

restringida a

$$\begin{cases} \dot{\theta}_i(t+dt) = \ f(\tau_i(t), \dot{\theta}_i(t)) \\ \dot{\theta}_i(t) - \dot{\theta}_{i_{max}} \leq 0 \\ \dot{\theta}_{i_{min}} - \dot{\theta}_i(t) \leq 0 \\ \theta_i(t) = \ \theta_{i_{inicial}} + \int_0^t \dot{\theta}_i(t)\, dt, \theta_{inicial} = \theta(t=0) \\ \theta_i(t) - \theta_{i_{max}} \leq 0 \\ \theta_{i_{min}} - \theta_i(t) \leq 0 \\ \tau_{i_{min}} \leq \tau_i(t) \leq \tau_{i_{max}} \end{cases}$$

per $i = 1:7$, on $(\theta_{real_i}(t) - \theta_i(t))^2$ és la funció objectiu que correspon a l'error de posició al quadrat al llarg del temps. S'utilitza aquesta funció objectiu ja que es vol controlar la

posició en tot moment. La funció $f$ correspon al model de predicció ARX no lineal obtingut que permet calcular la velocitat en qualsevol instant de temps a partir dels parells i de les velocitats en els instants anteriors. Els valors $\dot{\theta}_{i_{max}}$, $\dot{\theta}_{i_{min}}$, $\theta_{i_{max}}$, $\theta_{i_{min}}$, $\tau_{i_{min}}$ i $\tau_{i_{max}}$ són els valors mínims i màxims de velocitat, posició i parell per a cada articulació, ja especificats a l'apartat 4.3.

Aquestes equacions reescrites a temps discret són les següents:

$$\min_{\tau_i(KT)}(\theta_{real_i}(KT) - \theta_i(KT))^2$$

restringida a

$$\begin{cases} \dot{\theta}_i(KT+T) = f(\tau_i(KT), \dot{\theta}_i(KT)) \\ \dot{\theta}_i(KT) - \dot{\theta}_{i_{max}} \leq 0 \\ \dot{\theta}_{i_{min}} - \dot{\theta}_i(KT) \leq 0 \\ \theta_i(KT) = \theta_{i_{inicial}} + \sum_{k=0}^{K} \dot{\theta}(kT)T \\ \theta_i(KT) - \theta_{i_{max}} \leq 0 \\ \theta_{i_{min}} - \theta_i(KT) \leq 0 \\ \tau_{i_{min}} \leq \tau_i(KT) \leq \tau_{i_{max}} \end{cases}$$

per $i = 1:7$.

Anem a parlar ara del que anomenarem horitzó de predicció. Com ja s'ha comentat, el controlador utilitzarà el model de predicció i les dades corresponents a les posicions i les velocitats desitjades per predir quin parell cal aplicar a cada instant. Aquest procediment es farà mitjançant l'optimització presentada anteriorment que intentarà minimitzar l'error de posició al llarg del temps dins de l'horitzó de predicció. Aquest horitzó de predicció determinarà fins a quants instants en el futur es poden predir els parells a aplicar de manera que l'error de posició acumulat sigui mínim. És a dir, el controlador anirà calculant per a cada instant el parell a aplicar i a quina posició la simulació diu que anirà a parar

de manera que entre cada instant es desviï el mínim possible de la trajectòria desitjada. Aquest horitzó de predicció no es pot triar a ull ja que està relacionat amb la fiabilitat que ofereix el model per aproximar les dades sinó que s'ha de triar en funció del nombre d'instants que triga el model en començar-se a desviar de les dades originals a l'hora d'aproximar. Cal tenir cura amb l'horitzó de predicció que s'agafa ja que si es tria un horitzó molt gran i el model no és prou bon aproximador es poden anar acumulant errors.

Passem ara a descriure els resultats que s'han anat obtenint i els canvis que s'han anat introduint en el problema d'optimització.

Un cop implementats els programes d'optimització al MATLAB es van obtenir els primers resultats que van ser sorprenents, ja que l'optimització duta a terme permetia que les trajectòries es seguissin de manera que les posicions i velocitats simulades eren gairebé idèntiques a les reals. Però hi havia un inconvenient, ja que aquesta optimització permetia seguir les trajectòries molt fidelment però els parells que calculava com els necessaris per seguir-les eren molt sorollosos i presentaven una dispersió molt elevada a més a més d'increments de parell puntuals molt importants. Aquest fet no es podia passar per alt ja que unes variacions tant elevades en els parells a aplicar a les articulacions podia conduir fàcilment al deteriorament de les mateixes i per tant fer malbé el robot. És per això que es va considerar que s'havien d'introduir variacions en el disseny del controlador per mirar d'eliminar aquest fenomen.

Es va pensar en introduir un nou terme a la funció objectiu que permetés controlar els increments de parell entre instants per fer-los el menys grans possibles. Aquest terme havia de ser del tipus $\Delta\tau_i(KT) = \tau_i(KT) - \tau_i(KT - T)$, però per no tenir problemes d'ordres de magnitud entre ell i l'error de posició es van normalitzar els dos termes: l'error de posició $e_{pos_i}(KT) = \theta_{real_i}(KT) - \theta_i(t)$ es va dividir per l'error de posició màxim definit com $e_{pos_{i_{max}}} = \theta_{i_{max}} - \theta_{i_{min}}$ i l'increment de parell $\Delta\tau_i(KT) = \tau_i(KT) - \tau_i(KT - T)$ es va dividir per l'increment de parell màxim definit com $\Delta\tau_{i_{max}} = \tau_{i_{max}} - \tau_{i_{min}}$. Per evitar problemes amb el signe, ambdós termes es van elevar al quadrat i es va afegir una ponderació dels dos termes mitjançant dos pesos $p_1$ i $p_2$ que pot variar segons el terme que més interessi optimitzar.

En el nostre cas es desitjava el seguiment òptim de les trajectòries i afegint aquest segon terme s'aconseguia obtenir uns parells menys dispersos. Tot i que això podia significar

petits errors addicionals en el seguiment de les trajectòries, el pes corresponent a l'error de posició havia de ser més elevat per donar prioritat a l'objectiu de seguiment.

Per obtenir encara millors resultats, es va fer un últim canvi abans d'executar el programa d'optimització que té a veure amb el valor inicial d'iteració del parell. Com ja s'ha dit, per a cada instant de temps el robot intenta predir quin parell cal aplicar per tal de seguir la trajectòria desitjada i una cosa que es pot fer és introduir el valor amb el qual es desitja començar l'iteració, en lloc de començar-la amb el valor zero. Si per a cada instant es tria com a inici de l'iteració un valor adequat, el model mirarà de trobar la solució del problema no lineal al voltant d'aquest valor. Això indirectament ajuda a que les variacions de parell siguin menys pronunciades ja que el seguiment de trajectòries entre instants no requereix variacions de parell gaire brusques, i per tant la nova solució estarà a prop de l'anterior. En el nostre cas, el que s'ha fet és definir que el valor de parell amb el qual es comença a iterar en cada instant sigui el valor de parell aplicat a l'instant anterior. D'aquesta manera, el programa trobarà una solució de valor molt proper al que s'ha introduït i que s'aproparà molt més al parell real. A més a més, aquest fet d'introduir per a cada iteració el valor inicial de parell per a començar a calcular, servirà per agilitzar molt els càlculs ja que fins llavors el programa començava a iterar sempre a partir del valor zero per la qual cosa podia trigar molt a trobar la solució si aquesta s'allunyava molt de zero (o fins i tot podia no trobar-la).

L'addició d'aquests nous termes fa que el problema definit inicialment s'hagi d'ampliar de la següent manera:

$$\min_{\tau_i(KT)} \left( p_1 \left( \frac{\theta_{real_i}(KT) - \theta_i(KT)}{\theta_{i_{max}} - \theta_{i_{min}}} \right)^2 + p_2 \left( \frac{\tau_i(KT) - \tau_i(KT-T)}{\tau_{i_{max}} - \tau_{i_{min}}} \right)^2 \right)$$

restringida a

$$\begin{cases} \dot{\theta}_i(KT+T) = f(\tau_i(KT), \dot{\theta}_i(KT)) \\ \dot{\theta}_i(KT) - \dot{\theta}_{i_{max}} \leq 0 \\ \dot{\theta}_{i_{min}} - \dot{\theta}_i(KT) \leq 0 \\ \theta_i(KT) = \theta_{i_{inicial}} + \sum_{k=0}^{K} \dot{\theta}(kT)T \\ \theta_i(KT) - \theta_{i_{max}} \leq 0 \\ \theta_{i_{min}} - \theta_i(KT) \leq 0 \\ \tau_{i_{min}} \leq \tau_i(KT) \leq \tau_{i_{max}} \end{cases}$$

per $i = 1:7$.

Aquesta optimització s'ha dut a terme per a cadascuna de les articulacions del robot i per a les respectives trajectòries en les que es movien. L'addició dels pesos $p_1$ i $p_2$ ha permès suavitzar moderadament les variacions de parell al llarg de les trajectòries desitjades. Els resultats detallats i la representació gràfica de l'optimització es detallaran a l'apartat següent.

Un cop ampliada la funció de cost, es van obtenir nous resultats en els quals es continuava mantenint l'objectiu de seguiment de les trajectòries amb unes variacions de parell més suavitzades.

Encara que aquests resultats es van considerar com a molts bons, es va observar un cert caràcter oscil·lant al voltant dels parells sinusoidals que es volien aproximar. Es va realitzar, aleshores, un petit procés de filtratge dels parells obtinguts a partir de la predicció dels controladors. Per dur a terme aquest filtratge es va fer ús de la funció del Matlab `filter`, que permet utilitzar filtres de diferents ordres i amb paràmetres propis a definir.

Gràcies a aquests filtres es van acabar d'obtenir uns resultats encara més satisfactoris que els obtinguts anteriorment amb l'optimització ponderada amb pesos.

## 6.4 Resultats

Tal i com s'ha explicat detalladament a l'apartat anterior, per a controlar el sistema s'ha definit un disseny de controlador que s'ha anat millorant. Primer de tot es va definir només un únic terme a la funció objectiu i això va conduir a resultats bons per al seguiment de trajectòries però dolents pel que fa als parells que calia aplicar per a seguir-les. Aquests primers resultats es poden veure a continuació, on es poden observar les oscil·lacions pronunciades que es mencionaven a l'apartat anterior:

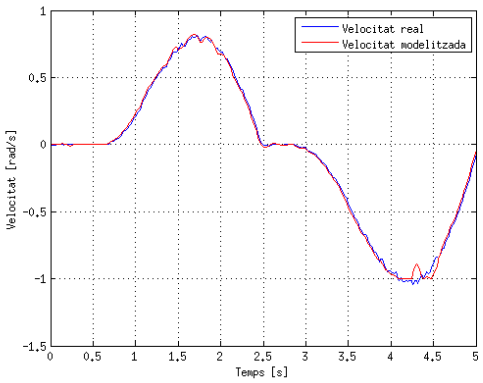


Figura 37: Comparació entre la velocitat real i la velocitat simulada amb el programa d'optimització.

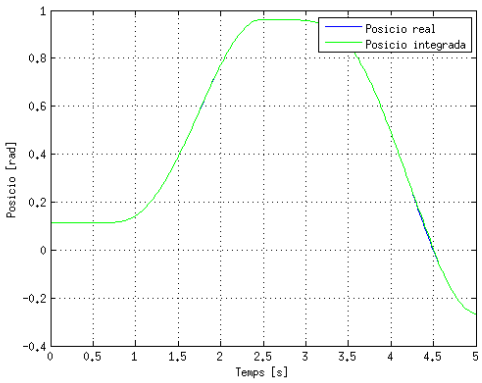


Figura 38: Comparació entre la posició real i la posició simulada amb el programa d'optimització.

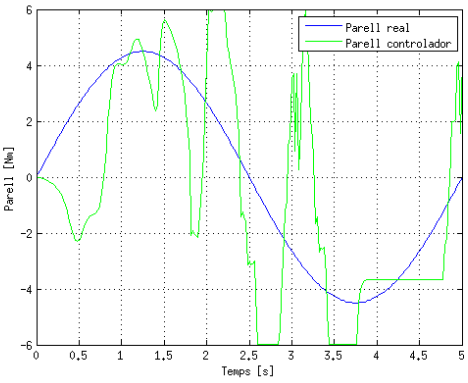


Figura 39: Comparació entre els parells reals i els parells predits pel controlador predictiu.

Com es pot observar a les Figures 37 i 38 , el seguiment de les trajectòries és molt bo ja que les comparacions entre les posicions i les velocitats reals amb les posicions

i les velocitats obtingudes en la simulació de la planta a partir de la predicció donen corbes gairebé idèntiques. En canvi, els parells obtinguts difereixen molt dels reals (vegi's la Figura 39). La diferència entre ells es deu a que les equacions amb les que s'està treballant són no lineals, pel que hi ha múltiples solucions. El model presentava, doncs, unes variacions brusques de parell no acceptables per la seguretat del WAM. Per arreglar-ho, es va introduir el segon terme a la funció objectiu suavitzant les variacions de parell entre instants i també el valor inicial d'iteració del parell en tot moment per a agilitzar els càlculs. Els resultats per a dues trajectòries de l'articulació 1 van ser els següents:

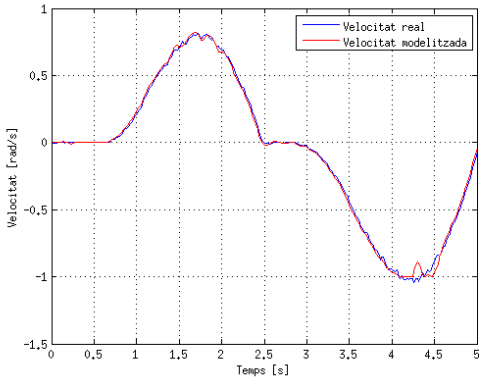


Figura 40: Comparació entre la velocitat real i la velocitat simulada amb el programa d'optimització

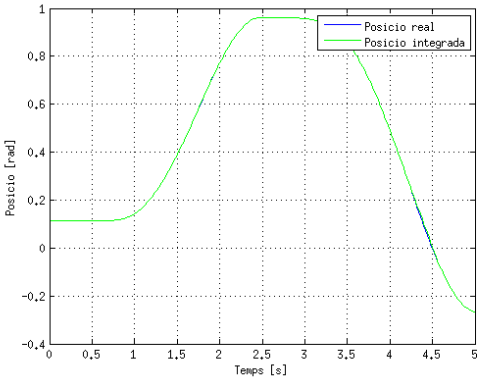


Figura 41: Comparació entre la posició real i la posició simulada amb el programa d'optimització

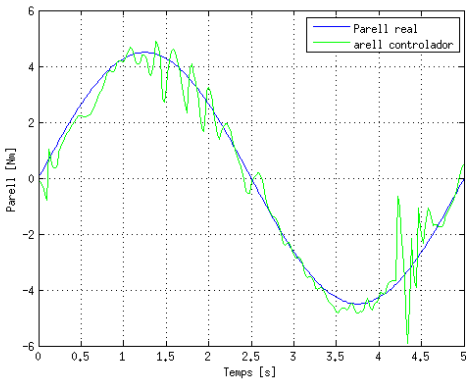


Figura 42: Comparació entre els parells reals i els parells predits pel controlador predictiu

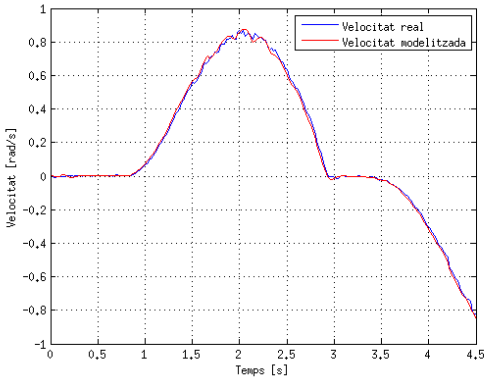


Figura 43: Comparació entre la velocitat real i la velocitat simulada amb el programa d'optimització

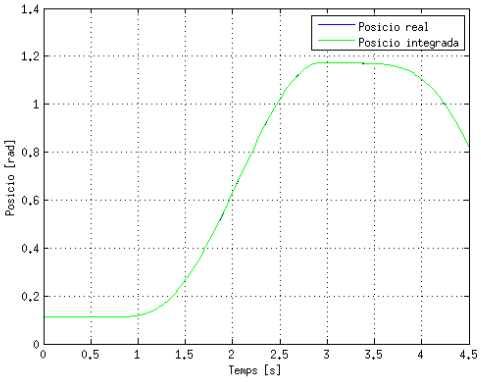


Figura 44: Comparació entre la posició real i la posició simulada amb el programa d'optimització

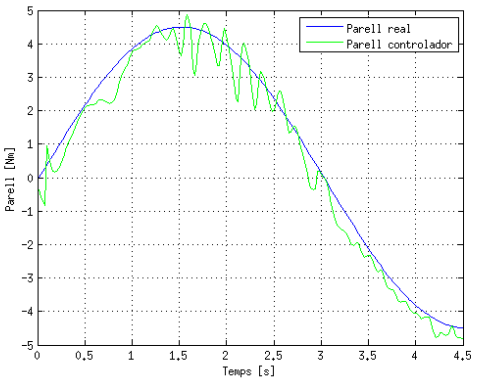


Figura 45: Comparació entre els parells reals i els parells predits pel controlador predictiu

Com es pot observar, els resultats obtinguts han estat molt millors que els anteriors ja que es continua garantint el seguiment de les trajectories com es pot observar en les Figures 40, 41, 43 i 44 però ara amb unes variacions de parell molt menys pronunciades com es mostra a les Figures 42 i 45.

Tot i la millora significativa que es va produir amb aquests nous resultats encara es podia observar que la corba de parell predit oscil·lava una mica al voltant de la corba sinusoidal que s'intentava aproximar. Es van filtrar aleshores les oscil·lacions de baixa freqüència presents per tal de que la nostra corba s'apropés més a la fonamental. Amb aquest filtratge es van obtenir resultats una mica millors tal i com es mostra a continuació:

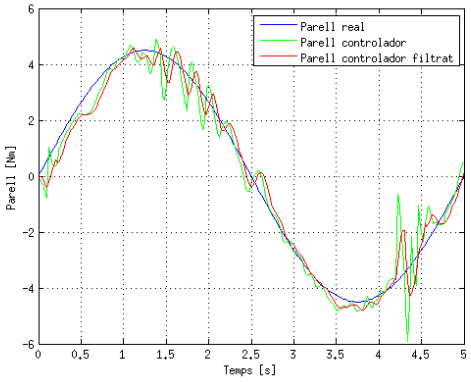


Figura 46: Comparació entre els parells reals i els parells predits pel controlador predictiu

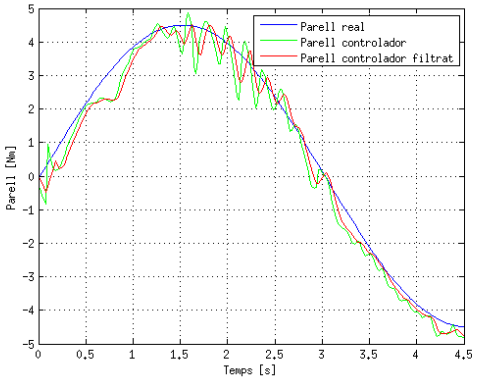


Figura 47: Comparació entre els parells reals i els parells predits pel controlador predictiu

Com es pot veure en les Figures 46 i 47, la nova corba de parell oscil·la amb una amplitud menor respecte de la corba fonamental sinusoidal i per tant els resultats es poden considerar encara millors.

Les figures que s'han mostrat en aquest apartat s'han obtingut prenent, com a model de simulació, el model ARX no lineal utilitzat com a model de predicció. S'han realizat els mateixos estudis utilitzant com a model de simulació el model Hammerstein-Wiener i s'han obtingut resultats similars.

A l'apartat següent s'explicarà el procés de validació realitzat per tal de validar els controladors predictius dissenyats per controlar el sistema.

## 6.5 Validació dels controladors

Un cop dissenyats els controladors predictius que es desitjaven i vistos els resultats que s'obtenien en simular-los amb el Matlab, es va començar a pensar en validar-los. La validació més lògica hauria estat fer proves de simulació amb el robot WAM directament. Això implicava la implementació en C++ de tots els programes de control i d'optimització així com dels propis models, però la magnitud i dificultat de tot això demanava un temps addicional que sobrepassava la durada de l'elaboració d'aquest projecte. Va ser per això que es van pensar noves maneres de validar els controladors predictius per mirar la seva precisió i qualitat.

La idea que es va pensar va ser la següent. El procediment a seguir per validar el robot ha consistit en primer lloc en calcular la predicció dels parells a aplicar en cada instant al llarg de certes trajectòries. Després s'agafen tots aquests valors i es guarden com una llista de valors. Tot seguit, aquesta llista s'introdueix al robot i s'observa com aquest es mou. Finalment, es comparen les trajectòries executades originalment amb les executades a partir de les llistes de valors de parell predits per part dels controladors.

A continuació s'exposen alguns dels motius que poden repercutir en que les trajectòries no es segueixin en la seva totalitat:

- Aquesta validació és un experiment en llaç obert a partir d'una llista de valors calculats en una simulació amb el Matlab del llaç tancat.
- La planta s'aproxima amb un model de simulació que no és totalment fidel al seu comportament i per tant les dades de sortida de la planta no seran iguals a les que sortirien del robot.
- El model de predicció tampoc aproxima el comportament del robot amb total fidelitat i per tant els parells predits poden ser lleugerament diferents als que caldria aplicar a la realitat.
- Els experiments executats no fan ús de cap realimentació i per tant no es poden actualitzar els estats del model de predicció tal i com es fa en les simulacions amb el Matlab.

De totes maneres, ja que en moltes de les articulacions els models de predicció i simulació aproximen amb un gran nivell d'exactitud el comportament dinàmic del robot, es pot donar bastant de fiabilitat a les dades corresponents als parells a aplicar calculats mitjançant els controladors predictius. Aquestes dades s'han introduït instant a instant al robot i s'ha observat que les trajectòries executades diferien poc de les reals. A continuació es mostra per a dues de les trajectòries executades amb l'articulació 1 les comparacions entre les trajectòries originals i les trajectòries executades amb els parells predits:

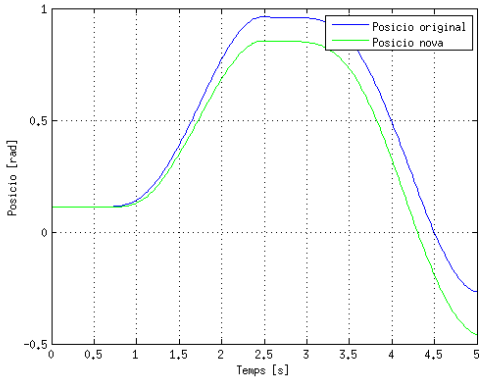


Figura 48: Comparació entre una trajectòria real i la trajectòria controlada.

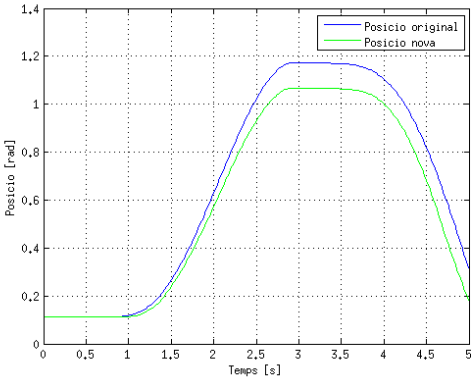


Figura 49: Comparació entre una trajectòria real i la trajectòria no controlada.

Com es pot observar a les Figures 48 i 49, els resultats són força bons, i les discrepàncies que existeixen es poden corregir tenint en compte els motius que abans s'han exposat.

# 7 Planificació temporal i costos

Aquest projecte s'ha dut a terme en el Laboratori de Manipulació i Percepció de l'Institut de Robòtica i Informàtica Industrial de la Universitat Politècnica de Catalunya (IRI-UPC) durant els mesos que van de febrer a juny de 2014. Essencialment s'ha treballat tots els matins de dilluns a dijous de 8.30 a 13.30 encara que quan ha calgut, sigui per poder fer reunions amb la directora i amb els dos investigadors amb què s'ha col·laborat o per intensificar alguna feina, s'ha ampliat l'horari de treball a altres dies i hores.

El cost del projecte es pot dividir en diverses partides degut a la diferent naturalesa dels elements que hi intervenen. Aquestes partides són:

- Recursos humans
- Amortització de l'equip
- Costos generals
- Beneficis industrials

En els següents apartats es detallaran els costos de cadascuna d'aquestes partides i el cost resultant total.

## 7.1 Recursos humans

En l'anàlisi del cost dels recursos humans es contempla el treball realitzat per l'estudiant per dur a terme el projecte així com el treball dut a terme per la directora del projecte i pels dos investigadors de l'IRI que li han donat suport.

Les hores de treball realitzades per l'estudiant s'han dedicat a la formació en els aspectes relacionats amb l'estudi realitzat, a l'estudi i pràctica amb manuals de MATLAB i C++, a l'anàlisi i definició del problema, a la programació de les funcions i dels programes que permeten obtenir els models dinàmics del robot així com la seva posterior validació, a la programació i disseny dels controladors necessaris juntament amb la conseqüent validació, a l'anàlisi dels resultats obtinguts i finalment a la redacció de la memòria.

Les hores de treball realitzades per la directora i els dos investigadors de l'IRI han estat dedicades sobretot a orientar, corregir, intervenir i validar el treball que l'estudiant ha anat realitzant al llarg de tot el projecte.

Els costos associats als recursos humans es mostren a la taula següent.

| Concepte | Hores utilitzades (h) | Cost per hora (€/h) | Cost (€) |
|---|---|---|---|
| Estudiant | 500 | 10 | 5000 |
| Investigadors | 150 | 30 | 4500 |
| Directora | 40 | 60 | 2400 |
| Total | | | 11900 |

Taula 1: Costos associats als recursos humans.

## 7.2 Amortització de l'equip

En aquest apartat s'analitzen els costos associats a l'ús d'equip o maquinària amb una vida útil més llarga que la durada del projecte.

En el Laboratori de Percepció i Manipulació de l'IRI, per l'elaboració d'aquest projecte s'han utilitzat:

1. Dos equips informàtics (el personal i el de control del WAM, que treballaven amb el sistema operatiu lliure Linux.

   En l'ordinador personal s'ha treballat amb una llicència de MATLAB ampliada perquè es necessitava la llibreria *System Identification Toolbox* que no ve per defecte amb la llicència de MATLAB estàndard i que ha permès realitzar les tasques de modelització, validació i interpretació dels resultats. A més a més, l'ordinador personal disposava dels programaris lliures LaTeXper la composició de textes científics i del C++.

   En l'ordinador de control s'ha treballat amb una llicència de MATLAB estàndard i amb el C++.

2. El robot WAM que es controlava directament des de l'ordinador de control i que disposava d'un dispositiu de seguretat per evitar accidents.

El cost d'ús d'aquests equips es valora com l'amortització lineal corresponent, tenint en compte el seu cost d'adquisició i de manteniment, la seva vida útil i les hores d'utilització. El cost de manteniment d'aquests aparells és d'un 8% de l'amortització que correspon a aquest projecte. Tot això es mostra a la taula 2:

| Concepte | Valor (€) | Cost de mant. (%) | Vida útil (h) | Amortització (€/h) | Ús (h) | Cost (€) |
|---|---|---|---|---|---|---|
| Ordinador personal | 1000.00 | 8 | 9000 | 0.12 | 360 | 43.20 |
| Ordinador WAM | 1000.00 | 8 | 9000 | 0.12 | 25 | 3.00 |
| Robot WAM | 105000.00 | 8 | 5000 | 22.68 | 25 | 567.00 |
| Dispositiu seguretat | 150.00 | 8 | 5000 | 0.03 | 25 | 0.75 |
| Total | | | | | | 613.95 |

Taula 2 Amortització del hardware.

Tenint en compte la informació de la taula anterior, el cost total associat a l'amortització de l'equip és de 613.95 €.

## 7.3 Costos generals

Els costos generals són els costos relatius a les feines d'administració, dels tècnics, les llicències de software, l'ús de l'electricitat i altres béns, com ara el material del laboratori per exemple, durant l'elaboració del projecte. L'article 131 del RGLCAP (*Reglamento General de la Ley de Contratos de las Administraciones Públicas*) fixa el valor d'aquestes costos en el 13% respecte el valor acumulat total fins ara. És a dir un 13% de 12513.95 , que suposa 1626.81€.

## 7.4 Beneficis industrials

Els beneficis industrials corresponen a la diferència entre el cost facturat total i el cost generat durant la realització del projecte, és a dir, als guanys que obté amb el projecte qui l'elabora. L'article 131 del RGLCAP (Reglamento General de la Ley de Contratos de las Administraciones Públicas) fixa el valor d'aquestes costos com el 6% respecte el valor acumulat total fins ara. És a dir un 6% de 14140.76, que suposa 848.45 €.

## 7.5 Cost total

Finalment en aquest apartat es reuniran els costos parcials de les diferentes partides per a calcular el cost resultant total:

| Concepte | Cost (€) |
|---|---|
| Recursos humans | 11900.00 |
| Amortització de l'equip | 613.95 |
| Costos generals | 1626.81 |
| Beneficis industrials | 848.45 |
| Total (sense IVA) | 14989.21 |
| Total (amb IVA) | 18136.94 |

Taula 1: Cost total del projecte.

El cost total del projecte és doncs de 18136.94 € amb l'IVA inclòs.

# 8 Impacte mediambiental

L'impacte mediambiental d'aquest projecte és practicament nul. En efecte,

D'una banda, s'ha treballat exclusivament amb un parell d'ordinadors i el braç robot WAM de l'IRI. Aquests estris igual que qualsevol aparell de l'IRI com els altres robots, les càmares de colors i profunditat, els sensors de força o qualsevol dels ordinadors amb els que els investigadors treballen habitualment, no tenen cap implicació directa medioambiental ja que només consumeixen energia elèctrica provinent de les xarxes d'alimentació. L'energia elèctrica de per si no contamina encara que sí ho fa la seva obtenció, però això queda fora de l'abast del nostre projecte.

D'altra banda, aquest projecte es veu molt poc afectat per l'impacte mediambiental provocat per la recollida dels residus d'aparells elèctrics i electrònics, ja que no hi ha hagut temps material per generar-ne i donat que aquesta recollida està totalment normalitzada a la UPC.

Les úniques accions, doncs, que s'ha pogut prendre són les que afavoreixen l'estalvi d'energia tal com indica el Pla d'Optimització Energètica de la UPC en el qual l'IRI participa.

D'aquesta manera es pot concloure que el nostre projecte s'ha pogut desenvolupar en un marc molt poc perjudicial pel medi ambient i això és un aspecte molt positiu que cal afegir a la seva consideració.

# 9 Línies futures d'estudi

Amb aquest projecte s'ha aconseguit una nova forma de controlar el robot WAM i una primera feina interessant a fer seria la implementació en llenguatge C++ dels models i dels programes d'optimització i control per tal que es puguin utilitzar en el WAM sempre que es desitgi. Però també és interessant demanar-nos si hi ha altres alternatives o tècniques de control que, un cop ben assimilat el projecte, es podrien explorar.

El control predictiu explicat en l'apartat 6 és una tècnica que dóna molt bons resultats però hi ha un factor que s'ha de tenir en compte i és que si el sistema amb el qual es treballa no és invariant al llarg del temps aleshores aquest control pot perdre precisió i qualitat amb el temps. En el nostre cas, el robot WAM pateix canvis a mesura que es va utilitzant, com ara el desgast, la sobretensió o l'afluixament dels seus cables, i aquests canvis poden modificar una mica el comportament global del robot. Si el model amb el qual es treballa no té en compte aquests canvis que van apareixent, és lògic pensar que la qualitat i precisió del control aniran disminuint a mesura que avanci el temps.

Una feina futura podria ser doncs millorar aquest control fent servir el que s'anomena *control adaptatiu*, apropiat per a qualsevol sistema que variï en el temps. El control adaptatiu és un tipus especial de control que consisteix en anar adaptant els paràmetres variables d'un procés amb la finalitat de mantenir un funcionament adequat del sistema al llarg del temps. S'utilitzen tècniques d'identificació de paràmetres, com ara mínims quadrats o tècniques similars, que permeten ajustar i anar actualitzant els models i els paràmetres bàsics del sistema que garanteixin un funcionament del sistema òptim en tot moment.

Mitjançant el control adaptatiu es podria fer més robust i precís el control que s'ha aconseguit en aquest projecte.

# 10 Conclusió

Pel que fa al projecte en si, un cop aquest ha finalitzat es pot afirmar que s'han assolit satisfactòriament el objectius que s'havien plantejat en un principi i que consten a la introducció d'aquesta memòria.

En primer lloc, s'ha realitzat un estudi exhaustiu del sistema amb el qual s'ha treballat per tal de definir correctament el problema a resoldre i mirar d'obtenir les solucions més adequades. Per això, s'han definit tant els objectius com el sistema de control amb el qual s'ha treballat així com l'espai de treball del robot, que inclou els rangs operacionals de valors que poden prendre els seus paràmetres. L'estudi realitzat ha estat suficientment ampli per saber com abordar el problema.

Pel que fa a l'obtenció del model dinàmic del robot, el resultat ha estat força positiu. Després de resoldre algunes incògnites com ara l'estructura que havien de presentar els models i el possible caràcter lineal o no lineal dels mateixos, s'han utilitzat tècniques d'identificació paramètriques per triar els models més bons aproximadors. En aquesta tria també s'ha tingut en compte que fos possible la seva implantació en codi en el WAM.

Aquest model dinàmic s'ha validat mitjançant els indicadors KPIs i el percentatge d'aproximació del model respecte de la realitat. Aquests indicadors han permès anar descartant models menys precisos que s'han anat obtenint.

Respecte del disseny dels controladors, un cop es van obtenir els models i es va veure que eren bons aproximadors, es va pensar en el control predictiu com a estratègia de control per al nostre sistema. Utilitzant la llibreria d'optimització del Matlab es va poder fer el control predictiu òptim que garantís un seguiment correcte de les trajectòries executades a partir d'uns parells que no presentessin variacions brusques. S'han explorat també altres alternatives variant els models de simulació però mantenint els de predicció fixos i els resultats han estat força satisfactoris.

Per últim, s'ha fet la validació dels controladors dissenyats comparant les trajectòries originals amb les executades a partir del parells obtinguts mitjançant el control predictiu. Aquesta validació ha donat uns resultats que s'han donat per bons.

Pel que fa als coneixements adquirits o ampliats durant el desenvolupament d'aquest projecte, aquests han estat força rellevants i formatius. Estic ben segur que aquests

coneixements i aquesta formació em seran molt útils tant en el Màster com en el meu futur com a Enginyer Industrial.

Finalment, l'elaboració d'aquest projecte en si ha estat, de nou, una experiència personal molt enriquidora i positiva. El fet d'haver-me d'enfrontar amb problemes reals i d'haver anat trobant solucions, sovint complexes, m'ha motivat molt a seguir treballant i a superar qualsevol entrebanc fins a arribar a complir els objectius que s'havien plantejat. Treballar en el Laboratori de Percepció i Manipulació de l'IRI, al costat dels seus investigadors i amb la generosa tutorització que he tingut, ha estat una oportunitat que no he desaprofitat i m'emporto la sensació d'haver treballat de valent en un projecte molt engrescador. De ben segur que aquesta experiència m'ajudarà en el futur a afrontar qualsevol projecte en el que participi.

# Bibliografia

[1] Colomé, A.; Pardo, D.; Alenyà, G. and Torras, C.: *External force estimation during compliant robot manipulation*, 2013 IEEE International Conference on Robotics and Automation, 2013, Karlsruhe, pp. 3535-3540.

[2] Lamport, Leslie: *LaTeX: A documentation prearation system*. 2nd ed, Addison Wesley, (1999).

[3] Ljung, Lennart: *System Identification - Theory For the User*, 2nd ed, PTR Prentice Hall, Upper Saddle River, N.J. (1999).

[4] Maaciejowski, J.M.: *Predictive Control with Constraints*, Prentice Hall, Great Britain, (2002).

[5] Mitrovic, Djorge; Nagashima, Sho; Klanke, Stefan; Matsubara, Takamitsu and Vijayakumar, Sethu: *Optimal Feedback Control for Anthropomorphic Manipulators*, IEEE international conference on robotics and automation (ICRA, 2010), Anchorage, Alaska, USA(2010).

[6] Planells, Antoni: *Identificació i test del model dinàmic d'un braç robot*. Treball dirigit del Grau en Enginyeria en Tecnologies Industrials fet a l'IRI amb una beca JAE INTRO 2013 del CSIC.

[7] Siciliano, Bruno; Sciavicco, Lorenzo; Villani, Luigi and Oriolo, Giuseppe: *Robotics modelling, planning and control*, Springer-Verlag London limited (2009).

[8] Vijayakumar, Sethu and Schaal, Stefan: *Locally Weighted Projection Regression : An O(n)Algorithm for IncrementalReal Time Learning in High Dimensional Space*, University of Southern California, Los Angeles, CA 90089-2520, USA.

**Altres fonts d'informació**:

[9] Biblioteca wikIRI, Institut de Robòtica i Informàtica Industrial de Catalunya. http://wiki.iri.upc.edu/index.php/Main-Page.

[10] C++ Tutorials, http://www.cplusplus.com/doc/tutorial/

[11] Manual del WAM,http://www.me.unm.edu/ starr/research/WAM-UsersGuide-AE-00.pdf.

[12] Mathworks documentation centre: System Identification Toolbox, http://www.mathworks.es/es/help/ident/index.html.

[13] Mathworks documentation centre: Optimitzation Toolbox, http://www.mathworks.es/es/help/optim/index.html.

[14] Mathworks documentation centre: Optimitzation Toolbox/Functions/Fmincon, http://www.mathworks.es/es/help/optim/ug/fmincon.html.

[15] Mathworks documentation centre: System Identification Toolbox/Nonlinear Model Identification/Nonlinear ARX Models, http://www.mathworks.es/es/help/ident/ug/identifying-nonlinear-arx-models.html.

[16] Mathworks documentation centre: System Identification Toolbox/Nonlinear Model Identification/ Hammerstein-Wiener Models, http://www.mathworks.es/es/help/ident/ug/identifying-hammerstein-wiener-models.html.

[17] Mathworks documentation centre: System Identification Toolbox/Linear Model Identification/Input-Output Polynomial Models, http://www.mathworks.es/es/help/ident/ug/identifying-input-output-polynomial-models.html.

[18] Pozo, Salvador: *Curso de C++* (2003), http://www.conclase.net

# Annexos

## Annex 1: Apunts teòrics

### Teorema de Nyquist-Shannon

El teorema del mostreig de Nyquist-Shannon és un teorema fonamental de la teoria de la informació, utilitzat en les telecomunicacions que va ser formulat per primera vegada per Harry Nyquist l'any 1928 i que va ser provat posteriorment per Claude E. Shannon l'any 1949.

Aquest teorema assegura que quan es mostreja un senyal de banda limitada, la freqüencia de mostratje $\omega_s$ ha de ser major que el doble de l'ampla B de banda ($\omega_{max}$=B) per a poder reconstruir el senyal original correctament. Aquest teorema es pot expressar de la següent manera:

$$\omega_s \geq 2 \cdot \omega_{max} \tag{16}$$

Aquest teorema es pot reescriure també en funció dels període de mostreig $T_s$ i periode d'oscil·lació mínim del senyal a mostrejar $T_{min}$:

$$T_s \leq \frac{T_{min}}{2} \tag{17}$$

Qualsevol procés de mostreig que no compleixi aquest teorema no garantirà una reconstrucció completa del senyal original amb la pèrdua d'informació que això comporti.

### Nombre de mostres per període d'oscil·lació

A partir del Teorema de Nyquist-Shannon anunciat anteriorment es pot definir el que s'anomena *nombre de mostre per període d'oscil·lació*. Aquest nombre correspon al nombre de vegades en que el sistema pren mostres del senyal portador durant cadascun dels períodes d'oscil·lació d'aquest. D'aquesta manera quantes més mostres es puguin prendre durant cada període del senyal millor serà la seva posterior reconstrucció. Es pot definir, aleshores, el nombre de mostres per període d'oscil·lació a partir dels períodes $T_s$ i $T_{min}$ o a partir de les pulsacions $\omega_s$ i $\omega_{max}$ com:

$$N = \frac{\omega_s}{\omega_{max}} = \frac{T_{min}}{T_s} \geq 2 \tag{18}$$

# Annex 2: Codi programes C++

## Programa iri_template.cpp

Aquest programa és el que s'ha utilitzat per executar les trajectòries en llaç obert mitjançant els parells sinusoidals que s'han definit prèviamnet per a cada articulació.

```
/*
 * iri_FeedForwardNonlinearControl.cpp
 *
 *
 * This Application uses a model of the inverse dynamics  u = f(q,q',q'').
 * to control the robot
 *
 * Adri'a Colom'e
 *
 * Usage : iri_template argv[1] argv[2] argv[3]
 *
 */

Ainclude <iostream>
Ainclude <string>
Ainclude <unistd.h>
Ainclude <barrett/units.h>
Ainclude <barrett/systems.h>
Ainclude <barrett/products/product_manager.h>
Ainclude <barrett/standard_main_function.h>
Ainclude <barrett/log.h>

Ainclude "iri_LearnedInverseDynamics.hpp"
Ainclude "iri_control_test.hpp"
Ainclude "dp_ja_type_system.hpp"
Ainclude "iri_splinetrajectory.hpp"
Ainclude "iri_frictionmodel.hpp"
Ainclude "iri_ma_system.hpp"
Ainclude "dp_DynamicsBaseSystem.hpp"
Ainclude "dp_ComputedTorqueMethod.hpp"


// this is to set functions by default
using namespace barrett;
using detail::waitForEnter;

// this function may be used to load data from files
template<size_t DOF, typename T>
int loadData(std::string filename, std::vector< T > & points);
```

```
// WAM MAIN function
template<size_t DOF>
int wam_main(int argc, char** argv, ProductManager& pm, systems::Wam<DOF>& wam) {

// load variable types and other things defined by barrett
        BARRETT_UNITS_TEMPLATE_TYPEDEFS(DOF);

// Check for input parameteres
// when calling function, we will write ./iri_template argv[1] argv[2] argv[3]
    if(argv[1]==NULL || argv[2]==NULL || argv[3]==NULL)
    {
        std::cout << "Missing some input!!" << std::endl;
        return 1; // this return 1 will stop the program if one input is missing.
    }
// exemple de donar un valor per defecte a un input al executar que pot no ser-hi
/*double guany;
if(argv[3]==NULL)
{
        guany=100.0;
}
else
{
        guany=argv[3];  // s'ha de tenir en compte que els argv son del tipus text!! per
                        passar a enter, per exemple, s'ha d'usar atoi(argv[3])
}*/

// the first input will be a set of points. We load its name and data
        std::string points_filename(argv[1]);
        std::vector<jp_type> qpoints_vector;
        int ok_data_1 = loadData<DOF,jp_type>(points_filename,qpoints_vector);
        int qpoints_size = (int)qpoints_vector.size();
        // if there is any problem with data, we stop the program:
        if(!ok_data_1)
        {
                std::cout << "Error during parameters/trajectory load, please verify the
                      files" << std::endl;
                wam.idle();
                pm.getSafetyModule()->waitForMode(SafetyModule::IDLE);
                return 1;
        }


// Gravity Compensation and Initialization
        wam.gravityCompensate();
        std::cout << "Wam is supposed to be @ home" << std::endl;
        waitForEnter();
```

```
// We create RAMP
        systems::Ramp my_ramp(pm.getExecutionManager());


// We create TRAJECTORY with given points
        splineTrajectory<DOF> my_traj(pm.getExecutionManager(),qpoints_vector,
                                      qpoints_vector[0],true);
                connect(wam.jpOutput,my_traj.jp_Input);
                connect(my_ramp.output,my_traj.timeValue);
/*splineTrajectoryGenerationMIMO<DOF,jp_type,jv_type,ja_type>

my_traj(pm.getExecutionManager(),qpoints_vector,qpoints_vector[0],speed,speed,true);
connect(my_ramp.output,my_traj.timeValue);*/



// We load friction parameters (make sure the parameters file is on the same
                                folder of execution!)
std::string PfileName("parametres2.txt");
                ifstream filep(PfileName.c_str());
                string linep;
                string valuep;
                stringstream issp;
                int Np,colnumber,linenumber,datanumber;
                Np=18;
                colnumber=0;
                linenumber=0;
                datanumber=0;
                Eigen::MatrixXd PAR;
                PAR.resize(11,Np);
                while ( filep.good() )
                {
                        while(getline(filep,linep)){
                                issp<<linep;
                                std::cout<<"\n iss="<<issp;
                                while ( getline(issp, valuep, ',') )
                                {
                                      colnumber=datanumber%Np;
                                      linenumber=datanumber/Np;
                                      datanumber++;
                                     PAR(linenumber,colnumber)=atof(valuep.c_str());

                                }
                                linep.clear();
                                issp.clear();
```

```
                    }
                }

// we create the friction model system with the loaded parameter matrix
frictionmodel<DOF> fric(pm.getExecutionManager(),PAR,"frictiomodel");
connect(my_traj.posOutput,fric.jpInput);
connect(my_traj.velOutput,fric.jvInput);
connect(my_traj.accelOutput,fric.jaInput);

// we create the coriolis and centripetal system
ComputedTorqueMethod<DOF> ctm(pm.getExecutionManager(),
pm.getConfig().lookup(pm.getWamDefaultConfigPath()));
connect(wam.jpOutput,ctm.jp_current_Input);
connect(wam.jvOutput,ctm.jv_current_Input);
connect(my_traj.accelOutput, ctm.ja_desired_Input); //we use desired acceleration
                                                      because its less noisy

// We define the PID controller
Eigen::VectorXd Pgains(DOF);

//Pgains << 300.0,500.0,200.0,100.0,30.0,2.0,0.5; // stiff PID controller
//Pgains << 100.0,150.0,80.0,40.0,10.0,2.0,0.25; // soft PID controller//
Pgains << 0.0,0.0,0.0,0.0,0.0,0.0,0.0; // this means no PID controller!!!
v_type Kp(Pgains);

controller_tests<DOF,jp_type,jv_type,jt_type> my_jpjvController(Kp);
connect(my_ramp.output,my_jpjvController.time);

/*State Grouper      : STATE Signal for position and velocity*/
systems::TupleGrouper<jp_type,jv_type> jpjv_StateGrouper("jp_jv_state");
connect(wam.jpOutput,jpjv_StateGrouper.template getInput<0>());
connect(wam.jvOutput,jpjv_StateGrouper.template getInput<1>());

/*Reference Grouper : GOAL Signal for position and velocity */
systems::TupleGrouper<jp_type,jv_type> jpjv_ReferenceGrouper("jp_jv_reference");
connect(my_traj.posOutput,jpjv_ReferenceGrouper.template getInput<0>());
connect(my_traj.velOutput,jpjv_ReferenceGrouper.template getInput<1>());

/* Controller Connections*/
connect(jpjv_StateGrouper.output,my_jpjvController.feedbackInput);

wam.supervisoryController.registerConversion(systems::makeIOConversion
(my_jpjvController.referenceInput,my_jpjvController.controlOutput));


// LOG SYSTEM
        const char BIN_FILE[] =    "./tmp/sequence_state.bin";
```

```
        typedef boost::tuple<double,jp_type,jv_type,jp_type,jv_type,ja_type,jt_type,
                             jt_type,jt_type> state_type;
        systems::PeriodicDataLogger<state_type> dl(pm.getExecutionManager(),
                            new log::RealTimeWriter<state_type>(BIN_FILE,
                                    pm.getExecutionManager()->getPeriod()));
        // Grouper for Log
        systems::TupleGrouper<double,jp_type,jv_type,jp_type,jv_type,ja_type,jt_type,
                              jt_type,jt_type> mi_TG("State-Action Grouper");
        connect(my_ramp.output,mi_TG.template getInput<0>());
        connect(wam.jpOutput,mi_TG.template getInput<1>());
        connect(wam.jvOutput,mi_TG.template getInput<2>());
        connect(my_traj.posOutput,  mi_TG.template getInput<3>());
        connect(my_traj.velOutput,  mi_TG.template getInput<4>());
        connect(my_traj.accelOutput,  mi_TG.template getInput<5>());
        connect(my_jpjvController.controlOutput,mi_TG.template getInput<6>());
        connect(fric.frictionOutput,mi_TG.template getInput<7>());
        connect(ctm.output,mi_TG.template getInput<8>());
        /* Ready for LOG */
        connect(mi_TG.output,dl.input);



// Get ready to execute and move to initial position
        std::cout << "Press [ENTER] for Initial Position " << std::endl;
        waitForEnter();
        wam.moveTo(qpoints_vector[0],0.75,0.75);



// Ready to start
        std::cout << "Press [ENTER] to Start" << std::endl;
        waitForEnter();


// Tell the robot which signal to follow
        wam.trackReferenceSignal(jpjv_ReferenceGrouper.output); // this will use our
                                            controller with [Pos;Vel] reference
//wam.trackReferenceSignal(my_traj.posOutput); //this will use default stiff PID
                                                controller with [Pos] reference

// Start moving!!
        my_ramp.start();
```

```
        std::cout << "Press [ENTER] to STOP" << std::endl;
        waitForEnter();
        my_ramp.stop();


// idle the wam.
        wam.idle();

        /* Saving data in CSV_FILE */

        const char CSV_FILE[] = "./tmp/log_PID_simple_model.txt";
        std::cout << "Saving... " << CSV_FILE << std::endl;
        dl.closeLog();
        log::Reader<state_type> lr(BIN_FILE);
        lr.exportCSV(CSV_FILE);


        /* Back to normal*/
        std::cout << "Press [Enter] to move HOME " << std::endl;
        waitForEnter();
        wam.moveHome();


        // Wait for the user to press Shift-idle
        pm.getSafetyModule()->waitForMode(SafetyModule::IDLE);

        return 0;
}



/* Auxiliary Function */
template<size_t DOF, typename T>
int loadData(std::string filename, std::vector < T > & points)
{
        BARRETT_UNITS_TEMPLATE_TYPEDEFS(DOF);

        std::cout << "Reading Files..." << std::endl;
        std::FILE * pFile=NULL;

        pFile = fopen (filename.c_str(), "r");
        if (pFile == NULL)
        {
                std::cout << "Error opening file" << std::endl;
                return false;
        }
```

```
        double aux;
        int index = 0;

        fpos_t curr_pos;
        fgetpos(pFile,&curr_pos);

        T temporal_vector;


        while (fscanf(pFile, "%lf ", &aux) != EOF )
        {
          fsetpos(pFile,&curr_pos);
          gsl_vector_fscanf(pFile, temporal_vector.asGslType());
          points.push_back(temporal_vector);
          std::cout << points[index] << std::endl;
          index++;
          fgetpos(pFile,&curr_pos);
        }
        waitForEnter();
        std::fclose(pFile);

        return 1;

}
```

### Programa iri_control_test.hpp

Aquest programa defineix el comportament del controlador encarregat d'enviar el valors dels parells a aplicar en cada instant a cadascuna de les articulacions del robot durant les trajectòries. Els parells en aquest cas són de forma sinusoidal on es pot ajustar la pulsació i amplitud de cadascun d'ells.

```
/*
 * iri_controller_tests.hpp
 *
 * System to compensate position and velocity
 *
 * Tau = Kp*ep + Kd*ev
 *
 * Difference with the standard PID_controller system (available in libbarrett) is that
 * it assumess desired velocity as ZERO.
 *
 * Here we offer a PD controller, where both (position and velocity) are references.
 *
```

```
 *
 *
 * Author : Diego Pardo
 *
 *
 *  ToDo : may be extended to be generic (Joint/Cartesian)
 *
 */

&ifndef IRI_SYSTEMS_CONTROLTEST_H
&define IRI_SYSTEMS_CONTROLTEST_H

&include <Eigen/Core>
&include <Eigen/SVD>

&include <barrett/detail/ca_macro.h>
&include <barrett/math/traits.h>
&include <barrett/systems/abstract/execution_manager.h>
&include <barrett/systems/abstract/controller.h>

template<size_t DOF,
                typename PosType,
                typename VelType,
                typename OutputType,
                typename MathTraits_pos = math::Traits<PosType>,
                typename MathTraits_vel = math::Traits<VelType> >
class controller_tests : public systems::Controller<boost::tuple<PosType,VelType>,
                                                    OutputType> {
        BARRETT_UNITS_TEMPLATE_TYPEDEFS(DOF);
public:
        systems::System::Input<double> time;

        typedef typename MathTraits_pos::unitless_type unitless_type_pos;
        typedef typename MathTraits_vel::unitless_type unitless_type_vel;

        explicit controller_tests(PosType Kp,const std::string& sysName =
                                "tptvControllerVar"):
                        systems::Controller<boost::tuple<PosType,VelType>,
                                OutputType>(sysName),
                        T_s(0.0), pos_error(0.0), vel_error(0.0), time(this),
            local_time(0.0),
                        kp(Kp), controlSignal(0.0), controlSignalLimit(500.0)
                        {
                                if (this->hasExecutionManager())
                                {
                            assert(this->getExecutionManager()->getPeriod()>0.0);
                                T_s = this->getExecutionManager()->getPeriod();
```

```
                        }
                        else {
                                T_s = 0.0;
                        }
                        J = gsl_matrix_alloc(6,7);

                        errint.resize(7);
                        errint_1.resize(7);
                        errint.fill(0.0);
                        errint_1.fill(0.0);

                        kgains.resize(7);
                        Kpcalc=kp;
                        index=0;
                        double temp[7] = {25.0, 20.0, 15.0, 15.0, 5.0, 5.0, 5.0};
                        for(int i =0;i<7;i++){
                                controlSignalLimit[i]=temp[i];
                                kgains(i)=1.0;
                        }


                        controlSignaleigen.resize(7);
                        controlSignaleigen.fill(0.0);
                        std::cout << "El Ts del tptvController : " << T_s <<
        std::endl;
                        std::cout << "Control Signal Limit -> " <<
        controlSignalLimit << std::endl;

                }

void setgains(Eigen::VectorXd ga){
        kgains=ga;
}

protected:

                bt_kinematics kinjac;
                gsl_matrix * J;
                Eigen::MatrixXd Je,Jpinv;
                Eigen::MatrixXd U,V,W,S,S0,KpJS,KdJS;
                Eigen::MatrixXd Spgain,Sdgain,Sigain;
                Eigen::VectorXd errint,errint_1,Kpcalc,Kup,Kdwn,vi,kp,kgains,
    controlSignaleigen,perr,verr,q,dq,dqd,qd;

                double normaux,local_time;
                int index;
```

```
        double T_s;
        PosType pos_error;
        VelType vel_error;

        int jacfound;
        /*unitless_type_pos kp;
        unitless_type_vel kd;*/
        OutputType controlSignal, controlSignalLimit;

        virtual void operate()
        {

                typedef MathTraits_pos MT_P;
                typedef MathTraits_vel MT_V;

                controlSignal = MT_P::zero();
                local_time = this->time.getValue();
                boost::tuple<PosType,VelType> state_values =
this->feedbackInput.getValue();
                boost::tuple<PosType,VelType> ref_values   =
this->referenceInput.getValue();

                q=boost::get<0>(state_values);
                qd=boost::get<0>(ref_values);

                dq=boost::get<1>(state_values);
                dqd=boost::get<1>(ref_values);



// Compute PD signal
                perr=qd-q;
                verr=dqd-dq;
                errint=errint_1+perr*0.002;
                errint_1=errint;
// Compute v-depending parameters
        Kpcalc.resize(7);


/*              // Compute gain matrix expressed in operational space
                                Spgain.resize(7,7);
                                Spgain.fill(0.0);
                                Sdgain.resize(7,7);
                                Sdgain.fill(0.0);
                                Sigain.resize(7,7);
                                Sigain.fill(0.0);
```

```
                        for (int k=0;k<7;k++){
                                Spgain(k,k)=Kpcalc(k);
                                // FER MAX I MIN PER SATURAR, mirar valor
        absolut!!!, regular exponent
                                Sdgain(k,k)=2.0*pow(Spgain(k,k),0.5)*0.2;
                                Sigain(k,k)=Sdgain(k,k)*0.25;
                        }

                        controlSignaleigen=Spgain*perr+Sdgain*verr+
    Sigain*errint;*/

//escriure aqui el valor dels joints usant local_time
                                double A1=5.0,A2=0.0,A3=0.0,A4=0.0,A5=0.0,
        A6=0.0,A7=0.0; // amplituds dels torques sinusoidals
                                double w1=0.1,w2=0.0,w3=0.0,w4=0.0,w5=0.0,
        w6=0.0,w7=0.0; // pulsacions dels torques sinusoidals
                        controlSignaleigen(0)=A1*sin(w1*local_time);
                        controlSignaleigen(1)=A2*sin(w2*local_time);
                        controlSignaleigen(2)=A3*sin(w3*local_time);
                        controlSignaleigen(3)=A4*sin(w4*local_time);
                        controlSignaleigen(4)=A5*sin(w5*local_time);
                        controlSignaleigen(5)=A6*sin(w6*local_time);
                        controlSignaleigen(6)=A7*sin(w7*local_time);

//aixo limita una mica els valors dels torques als joints
                        for(int iaux2=0;iaux2<7;iaux2++){
                                if(controlSignaleigen(iaux2)>
        controlSignalLimit[iaux2]){
                                        controlSignaleigen(iaux2)=
            controlSignalLimit[iaux2];
                                }else if(controlSignaleigen(iaux2)<
        -controlSignalLimit[iaux2]){
                                        controlSignaleigen(iaux2)=
            -controlSignalLimit[iaux2];
                                }
                        }

// aqui passem els valors del vector que tenim calculat al tipus de vector
que agafa la sortida
                        for(int iaux3=0;iaux3<7;iaux3++){
                                controlSignal[iaux3]=
        controlSignaleigen(iaux3);
                        }

                this->controlOutputValue->setData(&controlSignal);
                index++;
        }
```

```
        private:
                DISALLOW_COPY_AND_ASSIGN(controller_tests);

        public:
                EIGEN_MAKE_ALIGNED_OPERATOR_NEW_IF(MathTraits_vel::RequiresAlignment)

};


&endif /*IRI_SYSTEMS_CONTROLTEST_H */
```

### Programa iri_template.cpp

Aquest programa ha estat l'utilitzat per a dur a terme les validacions finals per mirar que els controladors dissenyats fossin els adequats.

```
/*
 * iri_FeedForwardNonlinearControl.cpp
 *
 *
 * This Application uses a model of the inverse dynamics  u = f(q,q',q'').
 * to control the robot
 *
 * Adri\'a Colom\'e
 *
 * Usage : iri_template argv[1] argv[2] argv[3]
 *
 */

&include <iostream>
&include <string>
&include <unistd.h>
&include <barrett/units.h>
&include <barrett/systems.h>
&include <barrett/products/product_manager.h>
&include <barrett/standard_main_function.h>
&include <barrett/log.h>

&include "iri_LearnedInverseDynamics.hpp"
&include "iri_control_test.hpp"
&include "dp_ja_type_system.hpp"
&include "iri_splinetrajectory.hpp"
&include "iri_frictionmodel.hpp"
&include "iri_ma_system.hpp"
&include "dp_DynamicsBaseSystem.hpp"
```

```
&include "dp_ComputedTorqueMethod.hpp"


// this is to set functions by default
using namespace barrett;
using detail::waitForEnter;

// this function may be used to load data from files
template<size_t DOF, typename T>
int loadData(std::string filename, std::vector< T > & points);


// WAM MAIN function
template<size_t DOF>
int wam_main(int argc, char** argv, ProductManager& pm, systems::Wam<DOF>& wam) {

// load variable types and other things defined by barrett
        BARRETT_UNITS_TEMPLATE_TYPEDEFS(DOF);

// Check for input parameteres
// when calling function, we will write ./iri_template argv[1] argv[2] argv[3]
    if(argv[1]==NULL || argv[2]==NULL || argv[3]==NULL)
    {
        std::cout << "Missing some input!!" << std::endl;
        return 1; // this return 1 will stop the program if one input is missing.
    }
// exemple de donar un valor per defecte a un input al executar que pot no ser-hi
/*double guany;
if(argv[3]==NULL)
{
        guany=100.0;
}
else
{
        guany=argv[3];  // s'ha de tenir en compte que els argv son del tipus text!! per
                        passar a enter, per exemple, s'ha d'usar atoi(argv[3])
}*/

// the first input will be a set of points. We load its name and data
        std::string points_filename(argv[1]);
        std::vector<jp_type> qpoints_vector;
        int ok_data_1 = loadData<DOF,jp_type>(points_filename,qpoints_vector);
        int qpoints_size = (int)qpoints_vector.size();
        // if there is any problem with data, we stop the program:
        if(!ok_data_1)
        {
                std::cout << "Error during parameters/trajectory load, please verify the
```

```
                files" << std::endl;
            wam.idle();
            pm.getSafetyModule()->waitForMode(SafetyModule::IDLE);
            return 1;
        }


// Gravity Compensation and Initialization
        wam.gravityCompensate();
        std::cout << "Wam is supposed to be @ home" << std::endl;
        waitForEnter();


// We create RAMP
        systems::Ramp my_ramp(pm.getExecutionManager());


// We create TRAJECTORY with given points
        splineTrajectory<DOF> my_traj(pm.getExecutionManager(),qpoints_vector,
                                      qpoints_vector[0],true);
            connect(wam.jpOutput,my_traj.jp_Input);
            connect(my_ramp.output,my_traj.timeValue);
/*splineTrajectoryGenerationMIMO<DOF,jp_type,jv_type,ja_type>

my_traj(pm.getExecutionManager(),qpoints_vector,qpoints_vector[0],speed,speed,true);
connect(my_ramp.output,my_traj.timeValue);*/



// We load friction parameters (make sure the parameters file is on the same folder
                              of execution!)
std::string PfileName("parametres2.txt");
            ifstream filep(PfileName.c_str());
            string linep;
            string valuep;
            stringstream issp;
            int Np,colnumber,linenumber,datanumber;
            Np=18;
            colnumber=0;
            linenumber=0;
            datanumber=0;
            Eigen::MatrixXd PAR;
            PAR.resize(11,Np);
            while ( filep.good() )
            {
                while(getline(filep,linep)){
                    issp<<linep;
```

```
                        std::cout<<"\n iss="<<issp;
                        while ( getline(issp, valuep, ',') )
                        {
                            colnumber=datanumber%Np;
                            linenumber=datanumber/Np;
                            datanumber++;
                            PAR(linenumber,colnumber)=atof(valuep.c_str());

                        }
                        linep.clear();
                        issp.clear();
                    }
                }


// We load FILE WITH TORQUES
std::string TfileName("torques.txt");
                ifstream filet(TfileName.c_str());
                string linet;
                string valuet;
                stringstream isst;
                colnumber=0;
                linenumber=0;
                datanumber=0;
                int Ntimesteps;
                Ntimesteps=100; // TONI: aqui nombre de punts que tens
                Eigen::MatrixXd TORQUES;
                TORQUES.resize(Ntimesteps,DOF);
                while ( filet.good() )
                {
                    while(getline(filet,linet)){
                        isst<<linet;
                        std::cout<<"\n iss="<<isst;
                        while ( getline(isst, valuet, ',') )
                        {
                            colnumber=datanumber%DOF;
                            linenumber=datanumber/DOF;
                            datanumber++;
                            TORQUES(linenumber,colnumber)=atof(valuet.c_str());

                        }
                        linet.clear();
                        isst.clear();
                    }
                }
                        std::cout<<"TORQUES=\n"<<TORQUES;
// we create the friction model system with the loaded parameter matrix
```

```
frictionmodel<DOF> fric(pm.getExecutionManager(),PAR,"frictiomodel");
connect(my_traj.posOutput,fric.jpInput);
connect(my_traj.velOutput,fric.jvInput);
connect(my_traj.accelOutput,fric.jaInput);

// we create the coriolis and centripetal system
ComputedTorqueMethod<DOF> ctm(pm.getExecutionManager(),
pm.getConfig().lookup(pm.getWamDefaultConfigPath()));
connect(wam.jpOutput,ctm.jp_current_Input);
connect(wam.jvOutput,ctm.jv_current_Input);
connect(my_traj.accelOutput, ctm.ja_desired_Input); //we use desired acceleration because
                                                    its less noisy

// We define the PID controller
Eigen::VectorXd Pgains(DOF);

//Pgains << 300.0,500.0,200.0,100.0,30.0,2.0,0.5; // stiff PID controller
//Pgains << 100.0,150.0,80.0,40.0,10.0,2.0,0.25; // soft PID controller//
Pgains << 0.0,0.0,0.0,0.0,0.0,0.0,0.0; // this means no PID controller!!!
v_type Kp(Pgains);

controller_tests<DOF,jp_type,jv_type,jt_type> my_jpjvController(Kp,TORQUES);
connect(my_ramp.output,my_jpjvController.time);

/*State Grouper      : STATE Signal for position and velocity*/
systems::TupleGrouper<jp_type,jv_type> jpjv_StateGrouper("jp_jv_state");
connect(wam.jpOutput,jpjv_StateGrouper.template getInput<0>());
connect(wam.jvOutput,jpjv_StateGrouper.template getInput<1>());

/*Reference Grouper : GOAL Signal for position and velocity */
systems::TupleGrouper<jp_type,jv_type> jpjv_ReferenceGrouper("jp_jv_reference");
connect(my_traj.posOutput,jpjv_ReferenceGrouper.template getInput<0>());
connect(my_traj.velOutput,jpjv_ReferenceGrouper.template getInput<1>());

/* Controller Connections*/
connect(jpjv_StateGrouper.output,my_jpjvController.feedbackInput);

wam.supervisoryController.registerConversion(systems::makeIOConversion
(my_jpjvController.referenceInput,my_jpjvController.controlOutput));


// LOG SYSTEM
        const char BIN_FILE[] =   "./tmp/sequence_state.bin";
        typedef boost::tuple<double,jp_type,jv_type,jp_type,jv_type,ja_type,jt_type,
                        jt_type,jt_type> state_type;
        systems::PeriodicDataLogger<state_type> dl(pm.getExecutionManager(),
                        new log::RealTimeWriter<state_type>(BIN_FILE,
```

```
                                    pm.getExecutionManager()->getPeriod()));
        // Grouper for Log
        systems::TupleGrouper<double,jp_type,jv_type,jp_type,jv_type,ja_type,jt_type,
                              jt_type,jt_type> mi_TG("State-Action Grouper");
        connect(my_ramp.output,mi_TG.template getInput<0>());
        connect(wam.jpOutput,mi_TG.template getInput<1>());
        connect(wam.jvOutput,mi_TG.template getInput<2>());
        connect(my_traj.posOutput,  mi_TG.template getInput<3>());
        connect(my_traj.velOutput,  mi_TG.template getInput<4>());
        connect(my_traj.accelOutput,  mi_TG.template getInput<5>());
        connect(my_jpjvController.controlOutput,mi_TG.template getInput<6>());
        connect(fric.frictionOutput,mi_TG.template getInput<7>());
        connect(ctm.output,mi_TG.template getInput<8>());
        /* Ready for LOG */
        connect(mi_TG.output,dl.input);


// Get ready to execute and move to initial position
        std::cout << "Press [ENTER] for Initial Position " << std::endl;
        waitForEnter();
        wam.moveTo(qpoints_vector[0],0.75,0.75);



// Ready to start
        std::cout << "Press [ENTER] to Start" << std::endl;
        waitForEnter();


// Tell the robot which signal to follow
        wam.trackReferenceSignal(jpjv_ReferenceGrouper.output); // this will use our
                              controller with [Pos;Vel] reference
//wam.trackReferenceSignal(my_traj.posOutput); //this will use default stiff PID
                              controller with [Pos] reference

// Start moving!!
        my_ramp.start();
        my_jpjvController.Start();

std::cout << "Press [ENTER] to STOP" << std::endl;
    waitForEnter();
    my_ramp.stop();
my_jpjvController.Stop();



// idle the wam.
```

```
        wam.idle();

        /* Saving data in CSV_FILE */

        const char CSV_FILE[] = "./tmp/log_PID_simple_model.txt";
        std::cout << "Saving... " << CSV_FILE << std::endl;
        dl.closeLog();
        log::Reader<state_type> lr(BIN_FILE);
        lr.exportCSV(CSV_FILE);


        /* Back to normal*/
        std::cout << "Press [Enter] to move HOME " << std::endl;
        waitForEnter();
        wam.moveHome();


        // Wait for the user to press Shift-idle
        pm.getSafetyModule()->waitForMode(SafetyModule::IDLE);

        return 0;
}



/* Auxiliary Function */
template<size_t DOF, typename T>
int loadData(std::string filename, std::vector < T > & points)
{
        BARRETT_UNITS_TEMPLATE_TYPEDEFS(DOF);

        std::cout << "Reading Files..." << std::endl;
        std::FILE * pFile=NULL;

        pFile = fopen (filename.c_str(), "r");
        if (pFile == NULL)
        {
                std::cout << "Error opening file" << std::endl;
                return false;
        }

        double aux;
        int index = 0;

        fpos_t curr_pos;
        fgetpos(pFile,&curr_pos);
```

```
        T temporal_vector;


        while (fscanf(pFile, "%lf ", &aux) != EOF )
        {
          fsetpos(pFile,&curr_pos);
          gsl_vector_fscanf(pFile, temporal_vector.asGslType());
          points.push_back(temporal_vector);
          std::cout << points[index] << std::endl;
          index++;
          fgetpos(pFile,&curr_pos);
        }
        waitForEnter();
        std::fclose(pFile);

        return 1;

}
```

### Programa iri_control_test.hpp

Aquest programa defineix el comportament del controlador encarregat d'enviar el valors dels parells a aplicar en cada instant a cadascuna de les articulacions del robot durant les trajectòries. Els parells en aquest cas s'obtenen a partir d'un document anomenat "torques.txt" que conté el valor dels parells a executar a cadascuna de les articulacions a cada instant de temps al llarg de les trajectòries.

```
/*
 * iri_controller_tests.hpp
 *
 * System to compensate position and velocity
 *
 * Tau = Kp*ep + Kd*ev
 *
 * Difference with the standard PID_controller system (available in libbarrett) is that
 * it assumess desired velocity as ZERO.
 *
 * Here we offer a PD controller, where both (position and velocity) are references.
 *
 *
 *
 * Author : Diego Pardo
 *
 *
 *  ToDo : may be extended to be generic (Joint/Cartesian)
```

```
 *
 */

&ifndef IRI_SYSTEMS_CONTROLTEST_H
&define IRI_SYSTEMS_CONTROLTEST_H

&include <Eigen/Core>
&include <Eigen/SVD>

&include <barrett/detail/ca_macro.h>
&include <barrett/math/traits.h>
&include <barrett/systems/abstract/execution_manager.h>
&include <barrett/systems/abstract/controller.h>

template<size_t DOF,
                typename PosType,
                typename VelType,
                typename OutputType,
                typename MathTraits_pos = math::Traits<PosType>,
                typename MathTraits_vel = math::Traits<VelType> >
class controller_tests : public systems::Controller<boost::tuple<PosType,VelType>,
                                                    OutputType> {
        BARRETT_UNITS_TEMPLATE_TYPEDEFS(DOF);
public:
        systems::System::Input<double> time;

        typedef typename MathTraits_pos::unitless_type unitless_type_pos;
        typedef typename MathTraits_vel::unitless_type unitless_type_vel;

        explicit controller_tests(PosType Kp, Eigen::MatrixXd TORQUES, const std::string&
                                sysName = "tptvControllerVar"):
                        systems::Controller<boost::tuple<PosType,VelType>,
            OutputType>(sysName),
                        T_s(0.0), pos_error(0.0), vel_error(0.0), time(this),
            local_time(0.0),
                        kp(Kp),torques(TORQUES),controlSignal(0.0),
            controlSignalLimit(500.0)
                        {
                                if (this->hasExecutionManager())
                                {
                                        assert(this->getExecutionManager()->getPeriod()
                > 0.0);
                                        T_s = this->getExecutionManager()->getPeriod();

                                }
                                else {
                                        T_s = 0.0;
```

```
                        }
                        J = gsl_matrix_alloc(6,7);

                        errint.resize(7);
                        errint_1.resize(7);
                        errint.fill(0.0);
                        errint_1.fill(0.0);
                        std::cout<<"TORQUES=\n"<<torques;
                        kgains.resize(7);
                        Kpcalc=kp;
                        index=0;
                        startflag=0;
                        Ntimesteps=torques.rows();
                        double temp[7] = {25.0, 20.0, 15.0, 15.0, 5.0, 5.0, 5.0};
                        for(int i =0;i<7;i++){
                                controlSignalLimit[i]=temp[i];
                                kgains(i)=1.0;
                        }

                        controlSignaleigen.resize(7);
                        controlSignaleigen.fill(0.0);
                        std::cout << "El Ts del tptvController : " << T_s <<
        std::endl;
                        std::cout << "Control Signal Limit -> " <<
        controlSignalLimit << std::endl;

                }

void setgains(Eigen::VectorXd ga){
        kgains=ga;
}
void Start(){
        startflag=1;
}

void Stop(){
        startflag=0;
}



protected:

                bt_kinematics kinjac;
                gsl_matrix * J;
                Eigen::MatrixXd Je,Jpinv;
                Eigen::MatrixXd U,V,W,S,S0,KpJS,KdJS;
```

```
            Eigen::MatrixXd Spgain,Sdgain,Sigain;
            Eigen::VectorXd errint,errint_1,Kpcalc,Kup,Kdwn,vi,kp,kgains,
controlSignaleigen,perr,verr,q,dq,dqd,qd;


            Eigen::MatrixXd torques;
            int Ntimesteps,startflag;

            double normaux,local_time;
            int index;
            double T_s;
            PosType pos_error;
            VelType vel_error;

            int jacfound;
            /*unitless_type_pos kp;
            unitless_type_vel kd;*/
            OutputType controlSignal, controlSignalLimit;

            virtual void operate()
            {

                    typedef MathTraits_pos MT_P;
                    typedef MathTraits_vel MT_V;

                    controlSignal = MT_P::zero();
                    local_time = this->time.getValue();
                    boost::tuple<PosType,VelType> state_values =
    this->feedbackInput.getValue();
                    boost::tuple<PosType,VelType> ref_values   =
    this->referenceInput.getValue();

                    q=boost::get<0>(state_values);
                    qd=boost::get<0>(ref_values);

                    dq=boost::get<1>(state_values);
                    dqd=boost::get<1>(ref_values);



    // Compute PD signal
                    perr=qd-q;
                    verr=dqd-dq;
                    errint=errint_1+perr*0.002;
                    errint_1=errint;
    // Compute v-depending parameters
```

```
                    Kpcalc.resize(7);


/*          // Compute gain matrix expressed in operational space
                                Spgain.resize(7,7);
                                Spgain.fill(0.0);
                                Sdgain.resize(7,7);
                                Sdgain.fill(0.0);
                                Sigain.resize(7,7);
                                Sigain.fill(0.0);
                                for (int k=0;k<7;k++){
                                        Spgain(k,k)=Kpcalc(k);
                                        // FER MAX I MIN PER SATURAR, mirar valor
                    absolut!!!, regular exponent
                                        Sdgain(k,k)=2.0*pow(Spgain(k,k),0.5)*0.2;
                                        Sigain(k,k)=Sdgain(k,k)*0.25;
                                }

                                controlSignaleigen=Spgain*perr+Sdgain*verr+
                Sigain*errint;*/

            //escriure aqui el valor dels joints usant local_time
                                        double A1=0.0,A2=0.0,A3=0.0,A4=0.0,A5=0.0,
                    A6=0.0,A7=0.09; // amplituds dels torques sinusoidals
                                        double w1=0.0,w2=0.0,w3=0.0,w4=0.0,w5=0.0,
                    w6=0.0,w7=0.33; // pulsacions dels torques sinusoidals
                                /*controlSignaleigen(0)=A1*sin(2.0*3.1416*
                                    w1*local_time);
                                controlSignaleigen(1)=A2*sin(2.0*3.1416*
                                    w2*local_time);
                                controlSignaleigen(2)=A3*sin(2.0*3.1416*
                                    w3*local_time);
                                controlSignaleigen(3)=A4*sin(2.0*3.1416*
                                    w4*local_time);
                                controlSignaleigen(4)=A5*sin(2.0*3.1416*
                                    w5*local_time);
                                controlSignaleigen(5)=A6*sin(2.0*3.1416*
                                    w6*local_time);
                                controlSignaleigen(6)=A7*sin(2.0*3.1416*
                                    w7*local_time);*/
                                if (index<Ntimesteps){
                                        controlSignaleigen(0)=torques(index,0);
                                        controlSignaleigen(1)=torques(index,1);
                                        controlSignaleigen(2)=torques(index,2);
                                        controlSignaleigen(3)=torques(index,3);
                                        controlSignaleigen(4)=torques(index,4);
                                        controlSignaleigen(5)=torques(index,5);
```

```
                                        controlSignaleigen(6)=torques(index,6);
                                }else{
                                        controlSignaleigen(0)=0.0;
                                        controlSignaleigen(1)=0.0;
                                        controlSignaleigen(2)=0.0;
                                        controlSignaleigen(3)=0.0;
                                        controlSignaleigen(4)=0.0;
                                        controlSignaleigen(5)=0.0;
                                        controlSignaleigen(6)=0.0;
                                }
        //aixo limita una mica els valors dels torques als joints
                                for(int iaux2=0;iaux2<7;iaux2++){
                                        if(controlSignaleigen(iaux2)>
                controlSignalLimit[iaux2]){
                                                controlSignaleigen(iaux2)=
                controlSignalLimit[iaux2];
                                        }else if(controlSignaleigen(iaux2)<
                -controlSignalLimit[iaux2]){
                                                controlSignaleigen(iaux2)=
                -controlSignalLimit[iaux2];
                                        }
                                }

        // aqui passem els valors del vector que tenim calculat al tipus de
    vector que agafa la sortida
                                for(int iaux3=0;iaux3<7;iaux3++){
                                        controlSignal[iaux3]=
                controlSignaleigen(iaux3);
                                }

                                this->controlOutputValue->setData(&controlSignal);
                if (startflag==1){
                                index++;
                }


                }

        private:
                DISALLOW_COPY_AND_ASSIGN(controller_tests);

        public:
                EIGEN_MAKE_ALIGNED_OPERATOR_NEW_IF(MathTraits_vel::RequiresAlignment)

};
```

```
&endif /*IRI_SYSTEMS_CONTROLTEST_H */
```

# Annex 3: Codi programes Matlab

## Programa modelhw.m

Aquest programa permet calcular la velocitat obtinguda a través del model Hammerstein-Wiener a partir dels parells que s'han anat executant en tot instant al llarg de la trajectòria. El model Hammerstein-Wiener està implementat en codi a partir dels seus paràmetres bàsics (vectors, matrius i funcions amb les que treballa).

```
clc
clear all
close all

%% Programa principal

joint=2;


load torq23A.mat
Vel=JV(:,joint);
if joint==1 || joint==4 || joint==7
    Torq=UC(:,joint);
elseif joint==2 || joint==3
    Torq=UC(:,2:3);
else
    Torq=UC(:,5:6);
end
T=size(JP,1);
Pos=JP(:,joint);

data=iddata(Vel,Torq,0.02);
t=data.SamplingInstants;

load modelhw2.mat
model=model2;

w=[];
Velocitat=[];
Estats=[];
Sortbloclineal=[];
Posint=[Pos(1,1)];
if joint==1 || joint==4 || joint==7
    Est=zeros(3,1); % Estat inicial del sistema lineal
else
    Est=zeros(6,2);
end
```

```
Posact=Pos(1,1);
for i=1:T
    if joint==1 || joint==4 || joint==7
        % Input nonlinearity
        Inputs=[Torq(i,1)];
        Inputnon=[];
        for l=2:(size(model.BP,2))
            if Inputs<=model.BP(1,l) && Inputs>=model.BP(1,l-1)
                sort=interpol(model.BP(1,l-1),model.BP(1,l),model.BP(2,l-1),...
                model.BP(2,l),Inputs);
            end
        end
        Inputnon=[Inputnon,sort];
        w=[w;Inputnon];
        % Bloc lineal
        x=model.C*Est;
        Est=model.A*Est+model.B*Inputnon';
        Sortbloclineal=[Sortbloclineal;x];
        %  Output nonlinearity
        Outputnon=x;
        for l=2:(size(model.bp,2))
            if Outputnon<=model.bp(1,l) && Outputnon>=model.bp(1,l-1)
                salida=interpol(model.bp(1,l-1),model.bp(1,l),model.bp(2,l-1),...
                model.bp(2,l),x);
            end
        end
        Velocitat=[Velocitat;salida];
        Posact=Posact+0.02*salida;
        Posint=[Posint;Posact];
    else
        salida=0;
        for j=1:2
            BP=cell2mat(model.BP(j));
            for l=2:(size(BP,2))
                if Torq(i,j)<=BP(1,l) && Torq(i,j)>=BP(1,l-1)
                    sort=interpol(BP(1,l-1),BP(1,l),BP(2,l-1),BP(2,l),Torq(i,j));
                end
            end
            x=cell2mat(model.C(j))*Est(:,j);
            Est(:,j)=cell2mat(model.A(j))*Est(:,j)+cell2mat(model.B(j))*sort;
            Outputnon=x;
            for l=2:(size(model.bp,2))
                if Outputnon<=model.bp(1,l) && Outputnon>=model.bp(1,l-1)
                    salida=salida+interpol(model.bp(1,l-1),model.bp(1,l),...
                    model.bp(2,l-1),model.bp(2,l),x);
                end
            end
```

```
        end

%          for l=2:(size(model.bp,2))
%              if Outputnon<=model.bp(1,l) && Outputnon>=model.bp(1,l-1)
%                  salida=interpol(model.bp(1,l-1),model.bp(1,l),...
%                    model.bp(2,l-1),model.bp(2,l),x);
%              end
%          end
        Velocitat=[Velocitat;salida];
        Posact=Posact+0.02*salida;
        Posint=[Posint;Posact];
    end
end

figure
plot(t,Vel,'b')
grid on
hold on
plot(t,Velocitat,'r')
ylabel 'Velocitat [rad/s]'
xlabel 'Temps [s]'
legend('Velocitat real','Velocitat modelitzada')


figure
plot(t,Pos,'b')
grid on
hold on
plot(t,Posint(1:size(Pos,1),1),'g')
ylabel 'Posicio [rad]'
xlabel 'Temps [s]'
legend('Posicio real','Posicio integrada')

%% C\'alcul error mitj\'a al quadrat i error mitj\'a en m\'odul

err=0;
err1=0;
nmost=size(Vel,1);
for i=1:nmost
    err=err+(Vel(i,1)-Velocitat(i,1))^2;
    err1=err1+abs(Vel(i,1)-Velocitat(i,1));
end
errquadmed=err/nmost;
errmodmed=err1/nmost;
```

### Programa modelnolineal.m

Aquest programa permet calcular la velocitat obtinguda a través del model Hammerstein-Wiener a partir dels parells que s'han anat executant en tot instant al llarg de la trajectòria. En aquest cas s'utilitza directament l'objecte "model" del Matlab i les comandes de simulació de models del *System Identification Toolbox*.

```
clc
clear all
close all

%% Load the sample data.

load torq1B.mat
idata=iddata(JV(:,1),UC(:,1),0.02);


%% load Hammerstein-Wiener model.

load modelhw1.mat

uNL = nlhw1.InputNonlinearity;
linModel = nlhw1.LinearModel;
yNL = nlhw1.OutputNonlinearity;

%% Compute the initial states that best fit the model response to the measured output.

% x0 = findstates(model1,idata);
x0=zeros(3,1);

%% Simulate the model using the estimated initial states.

% Input data for simulation
u = idata.u;
% Compute output of input nonlinearity
w = evaluate(uNL, u);
% Response of linear model to input w and zero initial conditions.
x = sim(linModel, w);
% Compute the output of the Hammerstein-Wiener model M
% as the output of the output nonlinearity estimator to input x.
ysim = evaluate(yNL, x);
% Compare low-level and direct simulation results.
t = idata.SamplingInstants;
figure
plot(t, idata.y, t, ysim, '.')
```

```
% ysim = sim(model1,data.u,'init',x0);

%% Compare ysim to output signal in z2:

% t = idata.samp;
% plot(t, ysim, t, idata.y)
% grid on
% legend('model','real')

%% construccio trajectoria modelada

Posint=[Pos(1,1)];
for j=1:size(Vel,1)
    Posactual=Posint(j,1);
    Posnova=Posactual+0.02*ysim(j,1);
    Posint=[Posint;Posnova];
end

figure
plot(t,Pos(:,1),'b')
hold on
plot(t,Posint(1:size(Vel,1),1),'g')
grid on
legend('real','model')

% Posint=[JP(1,1)];
% for j=1:size(JV,1)
%     Posactual=Posint(j,1);
%     Posnova=Posactual+0.02*ysim(j,1);
%     Posint=[Posint;Posnova];
% end
%
% figure
% plot(t,JP(:,1),'b')
% hold on
% plot(t,Posint(1:size(JV,1),1),'g')
% grid on
% legend('real','model')
```

### Funció costfunc.m

Aquesta funció permet definir correctament la funció de cost o també anomenada "funció objectiu" utilitzada per a la optimització amb la que treballa el controlador.

```
function cf = costfunc(tau,x0,pos0,model,Posdes,horitzo)
% pos0 \'es la posici\'o inicial a cada iteraci\'o
```

```
pos(1)=pos0;
error=((pos(1)-Posdes(1))/(model.pmax-model.pmin))^2;
inctorq=0;

for T=1:horitzo

    if T==1
        x(T)={x0};
    else
        u=cell2mat(x(T-1));
        x(T)={[v(T-1),u(1,1),tau(T-1),u(1,3),0,0]};  % J1

    end
    v(T)=modelnlarxsigmoid(cell2mat(x(T)),model.r,model.P,model.L,...
        model.Q,model.d,model.a_k,model.b_k,model.c_k,model.n);

    % Integraci\'o de la velocitat
    pos(T+1)=pos(T)+v(T)*model.T;

    % Error de posici\'o
    error=error+((pos(T+1)-Posdes(T+1))/(model.pmax-model.pmin))^2;

    % Increment torque
    reg=cell2mat(x(T));
    inctorq=inctorq+((tau(T)-reg(1,3))/(model.ub-model.lb))^2;

end

p1=10000;
p2=0.00005;

cf=p1*error+p2*inctorq;
```

### Funció const.m

Aquesta funció permet definir correctament les restriccions de desigualtat no lineals amb les que treballarà el nostre controlador a l'hora de dur a termes les tasques d'optimització.

```
function [c,ceq] = const(tau,x0,pos0,model,Posdes,horitzo)
% x0 'es l'estat actual del sistema a partir del qual farem el control
% predictiu

yyy=Posdes;
% pos0 \'es la posici\'o inicial a cada iteraci\'o
pos(1)=pos0;
```

```
for T=1:horitzo

    if T==1
        x(1)={x0};
    else
        u=cell2mat(x(T-1));
        x(T)={[v(T-1),u(1,1),tau(T-1),u(1,3),0,0]};  % J1
    end
    v(T)=modelnlarxsigmoid(cell2mat(x(T)),model.r,model.P,model.L,...
        model.Q,model.d,model.a_k,model.b_k,model.c_k,model.n);

    % Desigualtats velocitat
    c1(T)=model.vmin-v(T); % desigualtat inferior
    c2(T)=v(T)-model.vmax; % desigualtat superior


    % Desigualtats posici\'o
    pos(T+1)=pos(T)+v(T)*model.T;
    c3(T)=model.pmin-pos(T); % desigualtat inferior
    c4(T)=pos(T)-model.pmax; % desigualtat superior

end

ceq=[];

c=[c1;c2;c3;c4];
```

### Programa optimit.m

Aquest és el programa principal d'optimització que ha de permetre fer el seguiment precís de les trajectòries juntament amb el càlcul dels parells que s'han d'anar aplicant en cada instant per a garantir aquest seguiment.

```
clc
clear all
close all

joint=4;

load torq4B.mat

Posiciodes=JP(:,joint); % qr
Velocitatdes=JV(:,joint); % qp
if joint==1 || joint==4 || joint==7
    Torquedes=UC(:,joint);
elseif joint==2 || joint==3
```

```
    Torquedes=UC(:,2:3);
else
    Torquedes=UC(:,5:6);
end

data=iddata(Velocitatdes,Torquedes,0.02);
t=data.SamplingInstants;

load modelnlarxJ4.mat
% load modeltorq1.mat
horitzo=3;
if joint==1 || joint==4 || joint==7
    x0=zeros(1,4);
else
    x0=zeros(1,6);
end
tau0=0.5*ones(horitzo,size(Torquedes,2)); % mirar el joint
lb=model.lb*ones(horitzo,1);
ub=model.ub*ones(horitzo,1);
options2 = optimset('Display','on','LargeScale','on','MaxFunEvals',60000,...
    'MaxIter',3000,'TolFun',1e-6,'Algorithm','interior-point');

torq=[];
Posint=[Posiciodes(1,1)];
pos0=Posiciodes(1,1);
V=[];
Est=zeros(3,1); % mirar el joint
fv=[];
time=250;
for i=1:time
    i
    Posdes=Posiciodes(i:(i+horitzo),1);

    [tau,fval,exitflag,output]=fmincon('costfunc',tau0,...
        [],[],[],[],lb,ub,'const',options2,x0,pos0,model,Posdes,horitzo);

    exitflag;
    fv=[fv;fval];

    if i>15
        for j=1:size(tau0,2)
            tau0(:,j)=tau(1,j)*ones(horitzo,1);
        end
    end

    torq=[torq;tau(1,1)];
```

```
%
%       tau0=tau(1,1)*ones(horitzo,1);
%
      vel=modelnlarxsigmoid(x0,model.r,model.P,model.L,...
          model.Q,model.d,model.a_k,model.b_k,model.c_k,model.n); % NARX

%       [vel,Est]=modelo(tau(1,1),modelHW1,Est); % HW

      V=[V;vel];

      pos0=pos0+vel*model.T;
      Posint=[Posint;pos0];
      if joint==1 || joint==4 || joint==7
          x0=[vel,x0(1,1),tau(1,1),x0(1,3)]; % Regressor per la proxima iteracio
      else
          x0=[vel,x(1,1),tau(1,1),x(1,3),tau(1,2),x(1,5)];
      end
end

torqcor=[];
a = 1;
c = [1/5 1/5 1/5 1/5 1/5];
torqcor=filter(c,a,torq);

torques=zeros(i,7);
torques(:,joint)=torqcor;
dlmwrite('torques.txt',torques)

figure
plot(t(1:time,1),Posiciodes(1:time,1),'b')
hold on
grid on
plot(t(1:time,1),Posint(1:time,1),'g')
ylabel 'Posicio [rad]'
xlabel 'Temps [s]'
legend('Posicio real','Posicio integrada')

figure
plot(t(1:time,1),Velocitatdes(1:time,1),'b')
hold on
grid on
plot(t(1:time,1),V(1:time,1),'r')
ylabel 'Velocitat [rad/s]'
xlabel 'Temps [s]'
legend('Velocitat real','Velocitat modelitzada')

figure
```

```
plot(t(1:time,1),Torquedes(1:time,1),'b')
hold on
grid on
plot(t(1:time,1),torq(1:time,1),'g')
plot(t(1:time,1),torqcor(1:time,1),'r')
ylabel 'Parell [Nm]'
xlabel 'Temps [s]'
legend('Parell real','Parell controlador','Parell controlador filtrat')


figure
plot(fv)
```

### Programa modelnlarx.m

Aquest programa permet calcular la velocitat obtinguda a través del model ARX no lineal a partir dels parells que s'han anat executant en tot instant al llarg de la trajectòria. El model ARX no lineal està implementat en codi a partir dels seus paràmetres bàsics (vectors, matrius i funcions amb les que treballa).

```
clc
clear
close all


%% Model nonlinear ARX pel torque (amb bloc no lineal tipus sigmoidnet)

joint=2;

load modelnlarxJ2.mat % model a triar

load torq23A.mat % traject\'oria a triar

JP=JP(:,joint);
JV=JV(:,joint);
if joint==2 || joint==3
    UC=UC(:,2:3); % joints 2 i 3
elseif joint==5 || joint==6
    UC=UC(:,5:6); % joints 5 i 6
else
    UC=UC(:,joint); % joints 1, 4 i 7
end

data=iddata(JV,UC,0.02);
t=data.SamplingInstants;
```

```
Vmod=[];
Posint=[JP(1,1)];
if joint==1 || joint==4 || joint==7
    for i=1:size(JP,1)
        if i==1
            x=[0,0,0,0];
        else
            x=[Vmod(i-1,1),x(1,1),UC(i-1,1),x(1,3)];
        end
        v=modelnlarxsigmoid( x,model.r,model.P,model.L,...
            model.Q,model.d,model.a_k,model.b_k,model.c_k,model.n );
        Vmod=[Vmod;v];
        Posint=[Posint;Posint(i,1)+model.T*v];
    end
else
    for i=1:size(JP,1)
        if i==1
            x=[0,0,0,0,0,0];
        else
            x=[Vmod(i-1,1),x(1,1),UC(i-1,1),x(1,3),UC(i-1,2),x(1,5)];
        end
        v=modelnlarxsigmoid( x,model.r,model.P,model.L,...
            model.Q,model.d,model.a_k,model.b_k,model.c_k,model.n );
        Vmod=[Vmod;v];
        Posint=[Posint;Posint(i,1)+model.T*v];
    end
end

figure
plot(t,JV,'b')
hold on
grid on
plot(t,Vmod,'r')
ylabel 'Velocitat [rad/s]'
xlabel 'Temps [s]'
legend('Velocitat real','Velocitat modelitzada')


figure
plot(JP,'b')
hold on
grid on
plot(Posint,'g')
ylabel 'Posicio [rad]'
xlabel 'Temps [s]'
legend('Posicio real','Posicio integrada')
```

```
%% C\'alcul error mitj\'a al quadrat i error mitj\'a en m\'odul

err=0;
err1=0;
nmost=size(JV,1);
for i=1:nmost
    err=err+(JV(i,1)-Vmod(i,1))^2;
    err1=err1+abs(JV(i,1)-Vmod(i,1));
end
errquadmed=err/nmost; % mitjana dels quadrats dels errors
errmodmed=err1/nmost; % mitjana del m\'odul dels errors
```

### Funció modelnlarxsigmoid.m

Aquesta funció permet calcular la velocitat en un instant a partir del regressor en el propi instant tot utilitzant els paràmetres i funcions amb les que treballa el model ARX no lineal.

```
function [ v ] = modelnlarxsigmoid( x,r,P,L,Q,d,a_k,b_k,c_k,n )

v=(x-r)*P*L+d;
for j=1:n
    v=v+a_k(j,1)*(1/(exp(-((x-r)*Q*b_k(:,j)+c_k(1,j)))+1));
end
```

### Programa modelinit.m

Programa que permet crear les estructures de dades corresponents als paràmetres bàsics dels models Hammerstein-Wiener i ARX no lineals.

```
clear all
close all
clc

load modelnlarxJ1.mat

model.n=nlarx1.Nonlinearity.NumberOfUnits;
model.r=nlarx1.Nonlinearity.Parameters.RegressorMean;
model.Q=nlarx1.Nonlinearity.Parameters.NonLinearSubspace;
model.P=nlarx1.Nonlinearity.Parameters.LinearSubspace;
model.L=nlarx1.Nonlinearity.Parameters.LinearCoef;
model.b_k=nlarx1.Nonlinearity.Parameters.Dilation;
```

```
model.c_k=nlarx1.Nonlinearity.Parameters.Translation;
model.a_k=nlarx1.Nonlinearity.Parameters.OutputCoef;
model.d=nlarx1.Nonlinearity.Parameters.OutputOffset;
model.T=0.02;
model.vmin=-1.0;
model.vmax=1.0;
model.pmin=-2.6;
model.pmax=2.6;
model.ub=6.0;
model.lb=-6.0;

save modelnlarxJ1.mat
clear

load modeltorq1.mat

model.BP=nlhw1.InputNonlinearity.BreakPoints;
model.bp=nlhw1.OutputNonlinearity.BreakPoints;
lin=idss(nlhw1.LinearModel);
model.A=lin.A;
model.B=lin.B;
model.C=lin.C;
model.T=0.02;
model.pmin=-2.6;
model.pmax=2.6;
model.vmin=-2;
model.vmax=2;
model.lb=-6.0;
model.ub=6.0;

save modeltorq1.mat
clear
```

## Annex 4: Taules de resultats de les modelitzacions

En aquesta apartat es presenten les diferents taules dels indicadors KPIs i el percentatge d'aproximació per a cadascuna de les trajectòries executades amb les diferents articulacions i per als dos tipus de models no lineals obtinguts (ARX no lineals i Hammerstein-Wiener)

### Articulació 2

| Trajectòries | MSE NARX | MAE NARX | %fit NARX | MSE HW | MAE HW | %fit HW |
|---|---|---|---|---|---|---|
| Trajectòria A | 0.0012 | 0.0247 | 86.84 | 0.0021 | 0.0358 | 82.81 |
| Trajectòria B | 0.0027 | 0.0375 | 71.22 | 0.0035 | 0.0428 | 69.74 |
| Trajectòria C | 0.0013 | 0.0241 | 85.16 | 0.0022 | 0.0328 | 83.11 |
| Trajectòria D | 0.0016 | 0.0283 | 78.30 | 0.0018 | 0.0301 | 77.88 |
| Trajectòria E | 0.0013 | 0.0260 | 70.36 | 0.0009 | 0.0154 | 76.02 |
| Trajectòria F | 0.0027 | 0.0322 | 79.56 | 0.0031 | 0.0374 | 78.58 |
| Trajectòria G | 0.0043 | 0.0455 | 79.83 | 0.0016 | 0.0257 | 86.43 |
| Trajectòria H | 0.0046 | 0.0465 | 70.80 | 0.0039 | 0.0418 | 72.82 |
| Trajectòria I | 0.0040 | 0.0407 | 72.77 | 0.0038 | 0.0391 | 73.76 |
| Trajectòria J | 0.0043 | 0.0418 | 72.73 | 0.0041 | 0.0403 | 72.89 |

### Articulació 3

| Trajectòries | MSE NARX | MAE NARX | %fit NARX | MSE HW | MAE HW | %fit HW |
|---|---|---|---|---|---|---|
| Trajectòria A | 0.0011 | 0.0195 | 85.90 | 0.0017 | 0.0227 | 83.42 |
| Trajectòria B | 0.0026 | 0.0314 | 79.92 | 0.0031 | 0.0376 | 77.41 |
| Trajectòria C | 0.0019 | 0.0278 | 83.19 | 0.0018 | 0.0265 | 83.78 |
| Trajectòria D | 0.0027 | 0.0315 | 77.48 | 0.0032 | 0.0346 | 75.15 |
| Trajectòria E | 0.0037 | 0.0391 | 74.83 | 0.0027 | 0.0285 | 79.18 |
| Trajectòria F | 0.0024 | 0.0296 | 80.20 | 0.0030 | 0.0309 | 77.34 |
| Trajectòria G | 0.0018 | 0.0271 | 81.65 | 0.0013 | 0.0226 | 84.56 |
| Trajectòria H | 0.0031 | 0.0342 | 75.85 | 0.0037 | 0.0395 | 73.98 |
| Trajectòria I | 0.0033 | 0.0365 | 75.62 | 0.0028 | 0.0317 | 78.17 |
| Trajectòria J | 0.0029 | 0.0349 | 77.95 | 0.0032 | 0.0377 | 76.62 |

## Articulació 4

| Trajectòries | MSE NARX | MAE NARX | %fit NARX | MSE HW | MAE HW | %fit HW |
|---|---|---|---|---|---|---|
| Trajectòria A | 0.0084 | 0.0632 | 79.26 | 0.0082 | 0.0605 | 79.64 |
| Trajectòria B | 0.0089 | 0.0695 | 77.78 | 0.0058 | 0.0498 | 82.98 |
| Trajectòria C | 0.0067 | 0.0552 | 75.58 | 0.0070 | 0.0653 | 73.87 |
| Trajectòria D | 0.0084 | 0.0607 | 75.28 | 0.0102 | 0.0762 | 75.67 |
| Trajectòria E | 0.0085 | 0.0554 | 79.65 | 0.0119 | 0.0793 | 74.45 |
| Trajectòria F | 0.0096 | 0.0756 | 74.24 | 0.0113 | 0.0894 | 75.80 |
| Trajectòria G | 0.0124 | 0.0833 | 78.65 | 0.0097 | 0.0727 | 73.17 |
| Trajectòria H | 0.0083 | 0.0675 | 81.75 | 0.0093 | 0.0746 | 81.25 |
| Trajectòria I | 0.0084 | 0.0689 | 82.82 | 0.0090 | 0.0772 | 82.33 |
| Trajectòria J | 0.0090 | 0.0719 | 78.25 | 0.0085 | 0.0678 | 77.54 |

## Articulació 5

| Trajectòries | MSE NARX | MAE NARX | %fit NARX | MSE HW | MAE HW | %fit HW |
|---|---|---|---|---|---|---|
| Trajectòria A | 0.0018 | 0.0298 | 85.63 | 0.0029 | 0.0385 | 81.72 |
| Trajectòria B | 0.0024 | 0.0355 | 75.38 | 0.0033 | 0.0396 | 70.72 |
| Trajectòria C | 0.0017 | 0.0281 | 84.27 | 0.0021 | 0.0311 | 83.25 |
| Trajectòria D | 0.0023 | 0.0323 | 77.41 | 0.0016 | 0.0281 | 78.67 |
| Trajectòria E | 0.0037 | 0.0420 | 71.26 | 0.0025 | 0.0342 | 74.88 |
| Trajectòria F | 0.0032 | 0.0398 | 78.56 | 0.0026 | 0.0344 | 79.47 |
| Trajectòria G | 0.0043 | 0.0455 | 80.63 | 0.0013 | 0.0237 | 86.22 |
| Trajectòria H | 0.0041 | 0.0425 | 71.82 | 0.0036 | 0.0368 | 73.65 |
| Trajectòria I | 0.0033 | 0.0339 | 75.75 | 0.0041 | 0.0431 | 74.67 |
| Trajectòria J | 0.0042 | 0.0413 | 74.82 | 0.0040 | 0.0409 | 72.33 |

## Articulació 6

| Trajectòries | MSE NARX | MAE NARX | %fit NARX | MSE HW | MAE HW | %fit HW |
|---|---|---|---|---|---|---|
| Trajectòria A | 0.0013 | 0.0272 | 85.21 | 0.0016 | 0.0302 | 83.37 |
| Trajectòria B | 0.0039 | 0.0454 | 72.66 | 0.0031 | 0.0325 | 76.26 |
| Trajectòria C | 0.0014 | 0.0288 | 84.73 | 0.0017 | 0.0311 | 81.54 |
| Trajectòria D | 0.0024 | 0.0364 | 77.84 | 0.0019 | 0.0324 | 80.22 |
| Trajectòria E | 0.0023 | 0.0290 | 71.91 | 0.0021 | 0.0264 | 72.70 |
| Trajectòria F | 0.0035 | 0.0362 | 75.02 | 0.0037 | 0.0385 | 73.41 |
| Trajectòria G | 0.0038 | 0.0387 | 74.16 | 0.0030 | 0.0327 | 77.97 |
| Trajectòria H | 0.0032 | 0.0375 | 77.37 | 0.0034 | 0.0399 | 76.06 |
| Trajectòria I | 0.0043 | 0.0459 | 73.42 | 0.0034 | 0.0362 | 79.14 |
| Trajectòria J | 0.0038 | 0.0398 | 74.28 | 0.0045 | 0.0465 | 70.89 |

## Articulació 7

| Trajectòries | MSE NARX | MAE NARX | %fit NARX | MSE HW | MAE HW | %fit HW |
|---|---|---|---|---|---|---|
| Trajectòria A | 0.0093 | 0.0713 | 81.34 | 0.0099 | 0.0826 | 80.75 |
| Trajectòria B | 0.0138 | 0.0927 | 78.89 | 0.0089 | 0.0766 | 85.72 |
| Trajectòria C | 0.0078 | 0.0663 | 75.67 | 0.0081 | 0.0696 | 75.27 |
| Trajectòria D | 0.0095 | 0.0718 | 77.13 | 0.0113 | 0.0847 | 74.00 |
| Trajectòria E | 0.0086 | 0.0587 | 80.04 | 0.0098 | 0.0699 | 78.76 |
| Trajectòria F | 0.0085 | 0.0645 | 77.74 | 0.0102 | 0.0783 | 75.56 |
| Trajectòria G | 0.0113 | 0.0722 | 74.59 | 0.0086 | 0.0616 | 78.03 |
| Trajectòria H | 0.0076 | 0.0614 | 81.12 | 0.0087 | 0.0678 | 79.70 |
| Trajectòria I | 0.0081 | 0.0638 | 82.20 | 0.0088 | 0.0705 | 80.05 |
| Trajectòria J | 0.0092 | 0.0726 | 76.13 | 0.0086 | 0.0692 | 78.67 |